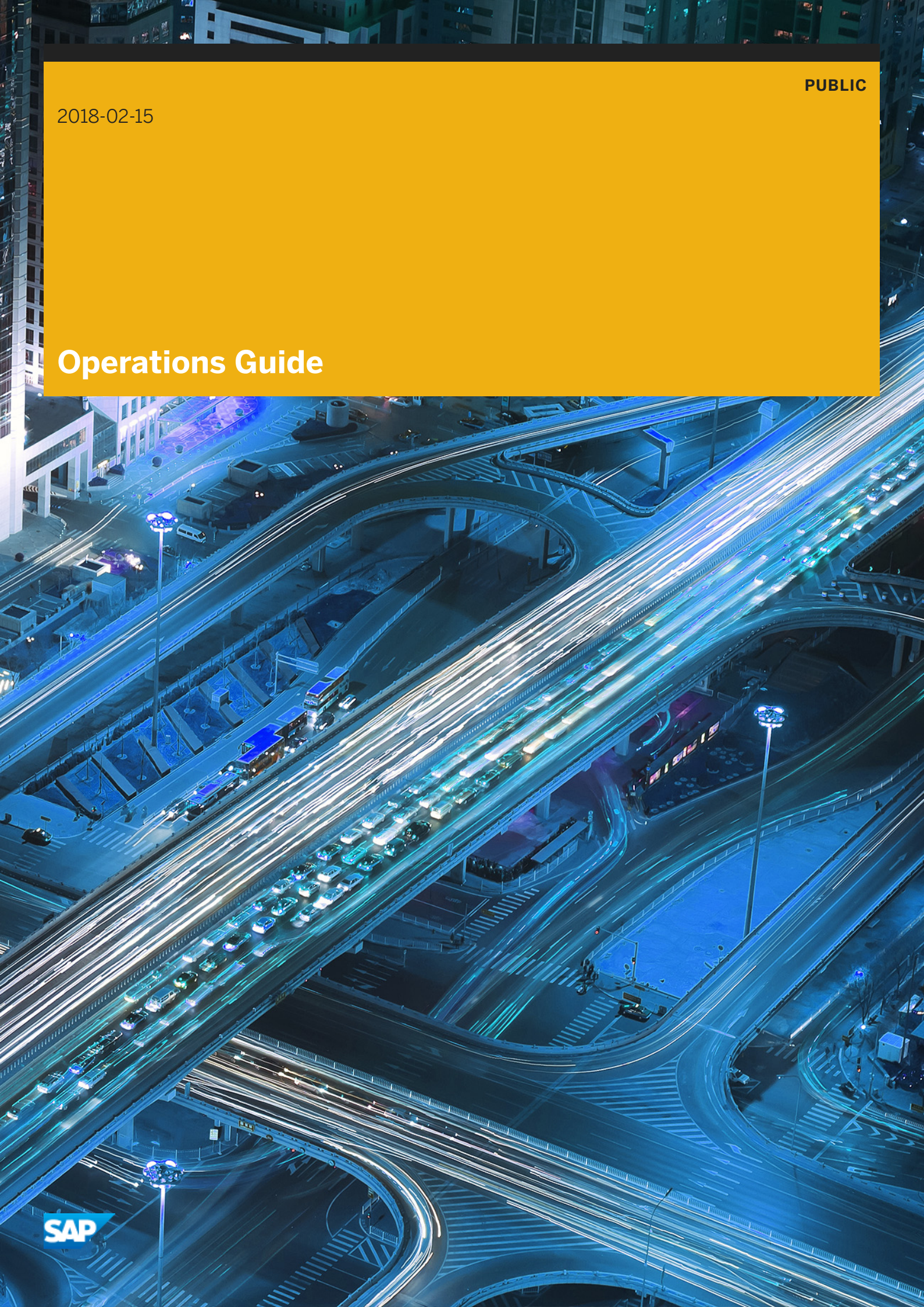
PUBLIC

2018-02-15

Operations Guide



Content

1 Understanding the Basic Concepts

This section provides an overview of the concepts and terms.

1.1 Runtime in Detail

For different customers separate resources (in terms of: memory, CPU, file system) of the cloud-based integration platform are allocated – although all customers might share the same hardware. This concept is also referred to as tenant isolation.

> **Note**
>
> A tenant represents the resources of the cloud-based integration platform allocated for a customer. Typically one tenant is defined for each customer connected to the platform.

At runtime, data to be exchanged between the involved customers is processed on a cluster of different virtual machines hosted in the SAP cloud, at which each virtual machine is assigned to the corresponding tenant allocated for the connected customer.

> **Note**
>
> A virtual machine (VM) is a software implementation of a machine that executes a program like a physical machine.

It is always made sure that the involved virtual machines are strictly separated from each other with regard to the related customers. In addition to that, each tenant uses a separate database schema which guarantees that data of different customers is strictly separated.

> **Note**
>
> The architecture guarantees that different tenants are unable to interfere and that physical resources of the cloud platform are partitioned per tenant.

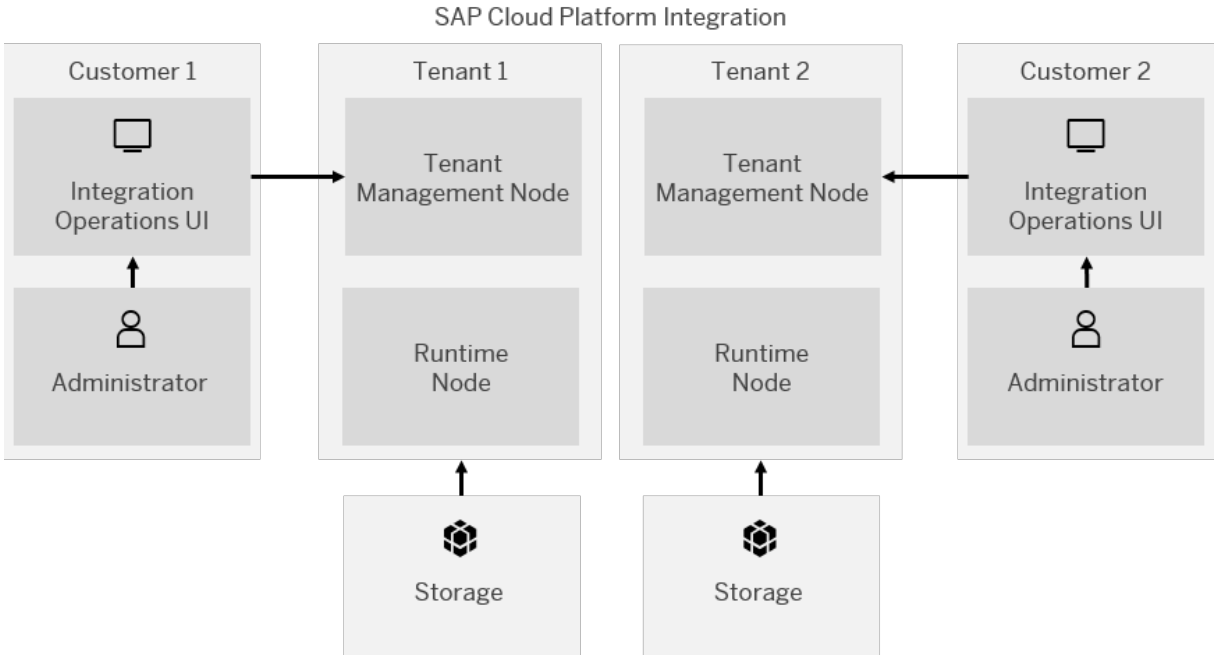
The runtime environment is composed of a cluster of virtual processes, where-by the message processing tasks for a tenant are performed within a dedicated Java Virtual Machine (JVM process or VM process). The individual processes are also referred to as nodes of the cluster.

A cluster is composed of different kinds of nodes.

Kind of Node	Purpose
Tenant management node	Performs tenant-specific management tasks like, for example, starting tenant-specific runtime nodes or deployment of artifacts like integration flows or keystores, for example.
Runtime node	Processes messages for a tenant. Services required for message processing like, for example, routing or mapping, are implemented as subsystems of the node.

> **Note**
>
> There is the option to set up multiple tenant management nodes for a tenant in order to implement failover scenarios and thus to ensure high availability. When one management node fails, one of the additional nodes can take over the tasks.

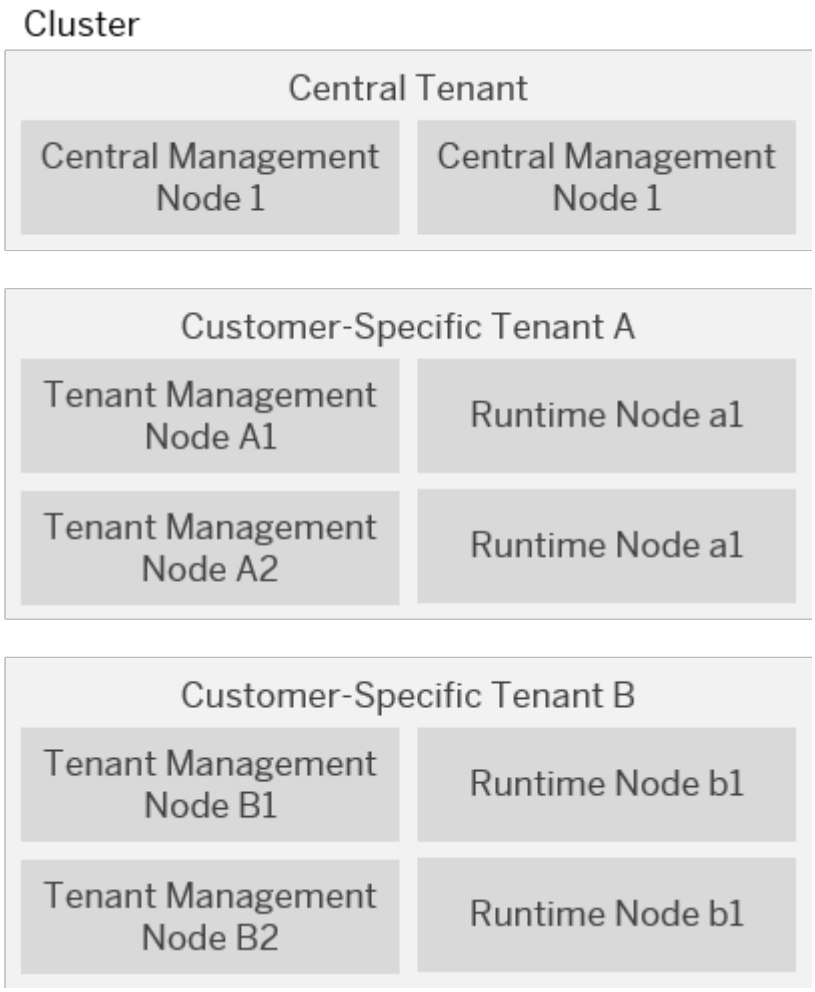
The following figure illustrates the general structure of a cluster.



A cluster for a tenant (shortly referred to as *tenant cluster*) is composed of one (or more) tenant management nodes and one or more runtime nodes.

Tenant clusters of different customers (customers) are strictly separated from each other and are unable to interfere.

In the Operations user interface (Node Explorer), the different kinds of nodes are arranged in the following way.



1.1.1 Virtual System Landscapes

There are different virtual system landscapes.

Landscapes, Virtual Servers and IP Addresses

You can access the following link to see the list of available landscapes and respective IP addresses: Landscape Hosts.

> **Note**
>
> The **IP addresses** are related to **outbound communication**, that means: for calls from the tenant to a receiver system. They are required by the customers to configure the firewall settings (*IP whitelisting*) to enable their system to connect to the cloud-based integration platform. Note that the table lists ranges of supported IP addresses rather than fixed IP addresses. This has the following advantage: In case of hardware problems at the cloud-based integration platforms side, the customer can quickly switch to

another machine (with another IP address) without the need to change the configuration in the back-end system.

Ports for Outbound Communication

When configuring an **outbound** channel (receiver adapter), make sure to use the following standard ports:

- Outbound **HTTP/HTTPS** connections: By default, **port 443 and all ports > 1024** are opened in the SAP Cloud Platform firewall.
  In case of any issues, open a ticket (component **LOD-HCI**).
- For **SFTP** connections, make sure that the SSH data channel between the SAP cloud platform and the SFTP server is opened. Use **port 22** at SAP Cloud Platform side.
- For **SMTP** connections, use **port 25**.

1.1.2 Stable URL

You can use this URL to access the tenant.

Management URL

Tenant	SSL Host	URL
Productive	HCI	https://<Account Short Name>-tmn.hci.<Landscape Host> For example, https://l0001-tmn.hci.eu1.hana.ondemand.com
Test	HCI	https://<Account Short Name>-tmn.hci.<Landscape Host>

Runtime URL

Tenant	SSL Host	URL
Productive	HCISBP	https://<Account Short Name>-ifl-map.hcisbp.<Landscape Host> For example, https://l0001-ifl-map.hcisbp.eu1.hana.ondemand.com
Test	HCISBT	https://<Account Short Name>-ifl-map.hcisbt.<Landscape Host>

Note

- In the URL, account short name refers to account detail mentioned in SAP email. You receive this emial after onboarding the tenant. Also,you can access the following link to see the list of available landscapes and respective IP addresses: Landscape Hosts.
- In SAP email, account short names begin with an alaphabet followed by 4-5 digits. For example, **A0001**.

2 Installing and Configuring the Tool

You need to install the Eclipse feature locally and then connect your locally installed Eclipse to SAP Cloud Platform Integration. In particular, you connect to the tenant cluster, which is that set of virtual machines allocated for your organization or company in SAP Cloud Platform.

Prerequisites

Your SAP contact has provided you with the URL of the tenant management node.

Context

The tenant management node is that virtual process of your tenant cluster that is responsible for the operation and management of your tenant cluster. It is the *connection point* for your organization or company to SAP Cloud Platform Integration.

Procedure

1. Install the features by following the instructions provided at: SAP Cloud Platform Integration Tools
2. Start Eclipse.
3. In the operation subsystem preferences (*Window* > *Preferences* > *SAP Cloud Platform Integration* > *Operations Server*) specify the URL provided to you by SAP.
4. To start working with the Integration Operations tooling, open the *Integration Operations* perspective in Eclipse (under *Window* > *Open Perspective*).

3 User Management for Cloud Integration

Users management tasks are required for different activities associated with the setup and operation of the cluster.

- To create users, you register at blogs.sap.com).
- Managing user-to-role assignments (using SAP Cloud Platform Cockpit).
  This task includes initial activities by the tenant administrator to define authorizations for people who are supposed to work on the tenant cluster.

Related Information

Creating a User for Cloud Integration [page 11]
Managing Users and Role Assignments [page 11]

3.1 Creating a User for Cloud Integration

For several tasks associated with SAP Cloud Platform Integration, you need a user account on blogs.sap.com.

Context

There are two types of users who can access SAP Cloud Platform Integration:

1. **S-User**: If you have an S-User ID, you are automatically a member of blogs.sap.com.
2. **P-User**: If you do not have an S-User ID, you can create a public user (P-User) account directly at blogs.sap.com .

After you have created a new user account, you can find your user ID in your account settings.

3.2 Managing Users and Role Assignments

You specify the members of the account and assign roles to them.

If this function is not available for your account, ask your SAP contact or create a ticket (component **`LOD-HCI`**) to activate it.

This task includes initial activities by the tenant administrator to assign roles to the users associated with all people who are supposed to work on the tenant cluster.

The task is sub dividied into the following main activities:

- Adding members to the account:
  With this step, you specify all users who should have assigned the same role like the tenant administrator In addition to that, you can also define all users who should have a *restricted* tenant administrator role (on the account). This specific role allows you to to assign application-specific roles to other users of the account (like, for example, the integration developer role).
- Defining authorizations for individual integration team members:
  With this step, you assign the actual application-specific roles to the integration team members (like, for example, the integration developer role)

To perform these steps, open the SAP Cloud Platform Cockpit using the S-user ID provided to you by SAP (in the mail that contains also the information on the account).

The URL of the SAP Cloud Platform Cockpit depends on the data center.

SAP Cloud Platform Cockpit URLs

Region	URL
Europe (Rot)	https://account.hana.ondemand.com/cockpit
US East (Ashburn)	https://account.us1.hana.ondemand.com/cockpit
Australia (Sydney)	https://account.ap1.hana.ondemand.com/cockpit

Related Information

Adding Members to an Account [page 12]
Defining Authorizations [page 14]
Overview of Authorization Groups [page 16]

3.2.1 Adding Members to an Account

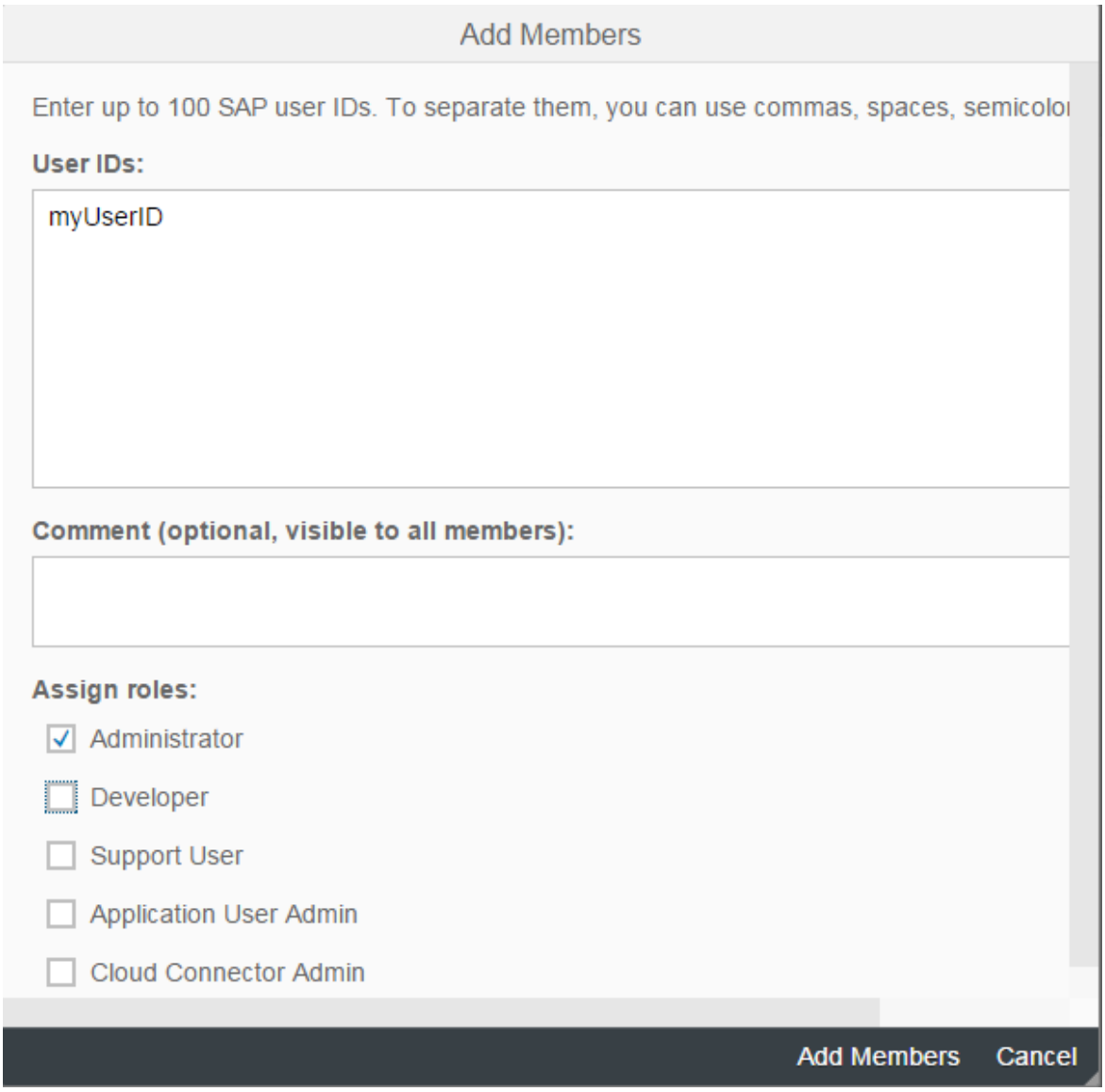
You specify the members of the account to define who is involved in an integration project.

Context

Perform the following steps.

Procedure

1. To specify all users who should get assigned the tenant administrator role, open the SAP Cloud Platform Cockpit using the S-user ID provided to you by SAP (in the mail that contains also the information on the account).
2. Choose *Members* > *Add Members*.
3. Enter the user ID.
4. Select the role which should be assigned to the user.



Assign role *Administrator* to the user who is supposed to have the full permissions of an administrator.

Assign role *Application User Admin* to the user who is supposed to have restricted administrator permissions.

> **Note**
>
> These roles are published by SAP Cloud Platform - they are **not** specific to any tasks related to integration projects.
>
> When you define more specific authorizations for integration team members, you assign other roles (as described in the corresponding section).

5. Choose *Add Members*.

Related Information

Defining Authorizations [page 14]
Overview of Authorization Groups [page 16]
Tasks and Required Roles [page 18]
Support Tasks [page 206]

3.2.2 Defining Authorizations

To authorize selected people to work on the account as part of the integration team in the context of SAP Cloud Platform Integration (for example, as integration developers), you assign roles to the associated users.

Context

Authorizations can also be given to users that are not associated with any integration team member (for example a user associated with a system that is to be authorized to send a message to Cloud Integration).

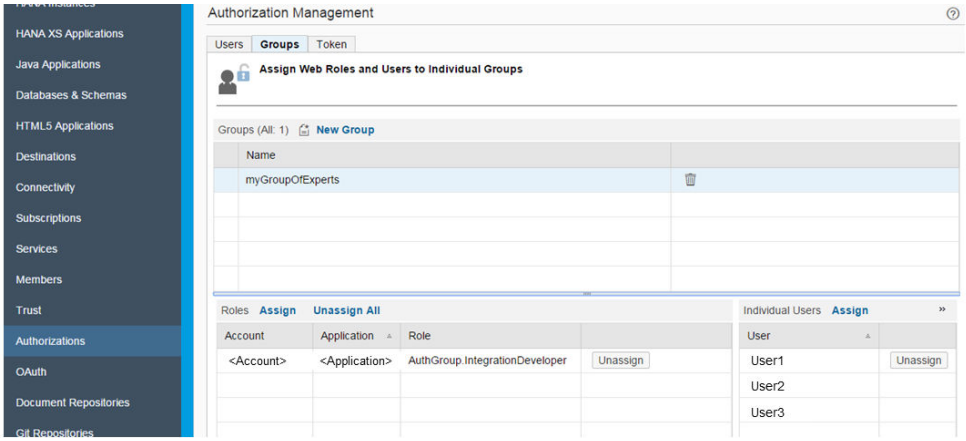
Perform the following steps for all users for which to assign authorizations.

> **Note**
>
> You can also define **user groups** and assign roles to user groups. That way, you can assign roles at once for all users who should to get identical permissions on the account.

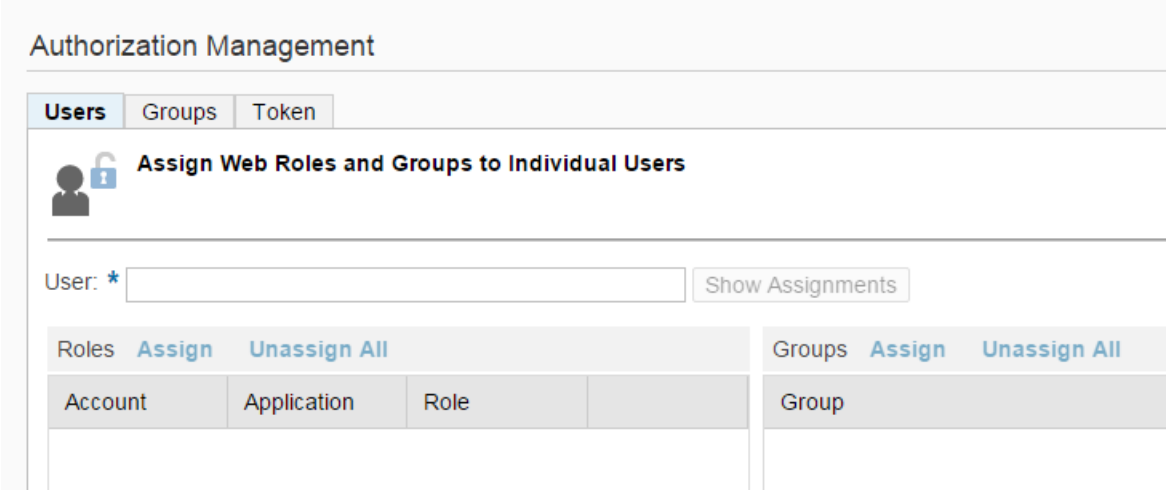
Procedure

1. In the navigation pane, choose *Security* > *Authorizations*.
2. If you like to first create a user group, on page *Authorization Management* choose the tab *Groups*.

   Enter a group name and add the relevant users to it (as shown in the figure).



3. On page *Authorization Management* enter the user group or user for which you like to assign authorization groups.

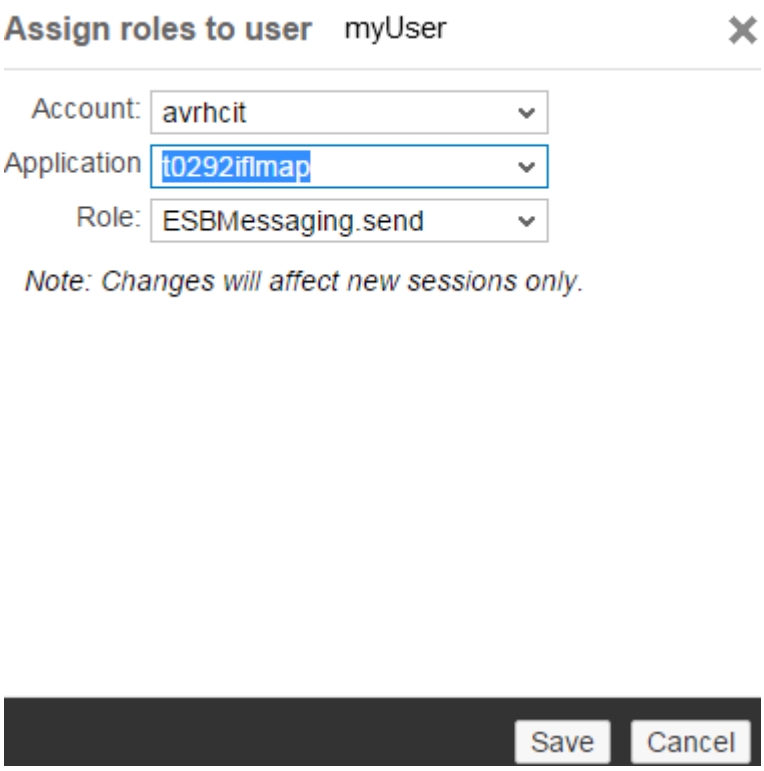


4. Choose *Assign*.
5. On page *Assign Roles to User <User Name>* choose the authorization group to be assigned.

We recomment that you always assign groups of roles (also referred to as **authorization groups**). There is a set of predefined authorization groups (beginning with *AuthGroup*) that cover the different asks associated with an integration project. Do **not** assign individual roles.

As *Application*, choose the one which ends with *tmn* (for tenant management node).

> ### Note
>
> There is one exception to the rule that only authorization groups should be assigned to users:
>
> In order to authorize a sender system to call a tenant (using HTTPS/basic authentication) and to get messages processed on the tenant, you need to assign to the associated technical user the specific role *ESBmessaging.send*. In that case, as *Application* choose the one which ends with *ifImap* (corresponding to a runtime node of the cluster which actually is in charge of processing the message).



6. Choose *Save*.

Related Information

Overview of Authorization Groups [page 16]

3.2.3 Overview of Authorization Groups

When you perform user management tasks using SAP Cloud Platform Cockpit, you find a set of pre-defined roles that you can assign to users of the account. According to the main tasks associated with integration projects, these roles are grouped in a set of authorization groups.

> **Note**
>
> You can choose among fine-granular roles and groups of roles (also referred to *authorization groups*, they begin with *AuthGroup*). Authorization groups cover the different tasks associated with an integration project. However, we recommend you to always assign authorization groups to users.
>
> In order to enable a sender system to process messages on a tenant using HTTPS/basic authentication, you need to assign to the associated user the role *ESBmessaging.send*. This role needs to be assigned to each (technical) user that is supposed to connect to Cloud Integration.

Authorization Groups

Authorization Group	Description
AuthGroup.BusinessExpert:	Enables a business expert to perform business tasks like, for example, examining the payload. This includes tasks like: • Monitoring integration flows and the status of integration artifacts • Reading the message payload and attachments
AuthGroup.Administrator	Enables the administrator of the tenant cluster (also referred to as the *tenant administrator*) to connect to a cluster and to perform administrative tasks on the cluster. This includes tasks like: • Monitoring integration flows and the status of integration artifacts • Deploying security content • Deploying integration content (such like integration flows, for example) • Deleting messages from transient data store
AuthGroup.IntegrationDeveloper	Enables an integration developer to connect to a cluster using Integration Designer and to display, download and deploy artifacts (for example, integration flows). This authorization group is required for accessing web tooling of Cloud Integration. This includes tasks like: • Monitoring integration flows and the status of integration artifacts • Deploying integration content (such like integration flows, for example)
AuthGroup.ReadOnly	Enables you to connect to a tenant cluster (from customer side) and to display nodes and node properties as well as to monitor messages. This authorization group enables you to access (read-only) the Data Store viewer.

Authorization Group	Description
AuthGroup.SystemDeveloper	Enables a system developer to perform tasks required for system support. This includes tasks like: • Monitoring integration flows and the status of integration artifacts • Restarting subsystems of the tenant cluster • Software development tasks on VMs of the tenant cluster This authorization group enables you to access (read-only) the Data Store viewer. **Note** System developer tasks are typically required in the support case by SAP experts who are supposed to perform tasks like debugging (for example) on the tenant cluster.

Note

Note the following comments to get a first overview of some of the authorization groups:

- The authorization group `<AuthGroup.Administrator>` is designed for the administrator at customer side who administers a (customer-specific) tenant management node.
- The authorization group `<AuthGroup.IntegrationDeveloper>` has to be assigned on TMN-level. Main difference between the authorization groups `<AuthGroup.IntegrationDeveloper>` and `<AuthGroup.Administrator>` is: the `<AuthGroup.Administrator>` group allows in addition to deal with security artifacts (deployment of keystore files, for example).

Related Information

Tasks and Required Roles [page 18]

3.2.3.1 Tasks and Required Roles

The following table provides an overview of which roles are required in order to accomplish the various tasks related to SAP Cloud Platform Integration. It is also indicated in how far the tasks and roles are relevant for the main persona and authorization groups defined for Cloud Integration.

Tasks and Roles

Area	Task	Role	Integration Developer (AuthGroup.IntegrationDeveloper)	Business Expert (AuthGroup.BusinessExpert)	Supporter/ System Developer (AuthGroup.SystemDeveloper)	Tenant Administrator (AuthGroup.Administrator)
Discover	View packages	WebToolingCatalog.Overview-Read	assigned	assigned	assigned	assigned
Discover	View package artifacts	WebToolingCatalog.Overview-Read WebToolingCatalog.Details-Read	assigned	assigned	assigned	assigned
Discover	Copy package to workspace	WebToolingCatalog.Overview-Read WebTooling-Workspace.Write	assigned	n.a.	n.a.	n.a.
Design	View packages and package artifacts	WebTooling-Workspace.Read	assigned	assigned	assigned	assigned
Design	Create, edit, import, export, delete package with its artifacts	WebTooling-Workspace.Read WebTooling-Workspace.Write	assigned	n.a.	n.a.	n.a.
Design	Update package	WebTooling-Workspace.Read WebTooling-Workspace.Write	assigned	n.a.	n.a.	n.a.

Area	Task	Role	Integration Developer (AuthGroup.IntegrationDeveloper)	Business Expert (AuthGroup.BusinessExpert)	Supporter/ System Developer (AuthGroup.SystemDeveloper)	Tenant Administrator (AuthGroup.Administrator)
Design	Configure artifacts (integration flows and value mappings)	WebTooling-Workspace.Read WebTooling.IntegrationFlow-Configure	assigned	n.a.	n.a.	n.a.
Design	Deploy/undeploy artifacts	WebTooling-Workspace.Read NodeManager.read GenerationAndBuild.generationandbuildcontent NodeManager.deploycontent	assigned	n.a.	n.a.	assigned
Design	Export Package for transport	WebTooling-Workspace.Read IntegrationContent.Transport	n.a.	n.a.	n.a.	n.a.
Design	Import package from transport	WebTooling-Workspace.Read WebTooling-Workspace.Write IntegrationContent.Transport	n.a.	n.a.	n.a.	n.a.

Area	Task	Role	Integration Developer (AuthGroup.IntegrationDeveloper)	Business Expert (AuthGroup.BusinessExpert)	Supporter/ System Developer (AuthGroup.SystemDeveloper)	Tenant Administrator (AuthGroup.Administrator)
Design	Update Package from transport	WebTooling-Workspace.Read WebTooling-Workspace.Write IntegrationContent.Transport	n.a.	n.a.	n.a.	n.a.
Monitor	View Monitor Overview	IntegrationOperation-Server.read NodeManager.read	assigned	assigned	assigned	assigned
Monitor	View message processing logs	IntegrationOperation-Server.read	assigned	assigned	assigned	assigned
Monitor	View tasks	IntegrationOperation-Server.read NodeManager.read	assigned	assigned	assigned	assigned
Monitor	View tail log	IntegrationOperation-Server.read NodeManager.read	assigned	assigned	assigned	assigned
Monitor	View deployed artifact list	IntegrationOperation-Server.read NodeManager.read	assigned	assigned	assigned	assigned

Area	Task	Role	Integration Developer (AuthGroup.IntegrationDeveloper)	Business Expert (AuthGroup.BusinessExpert)	Supporter/ System Developer (AuthGroup.SystemDeveloper)	Tenant Administrator (AuthGroup.Administrator)
Monitor	View deployed integration flow in graphical editor	IntegrationOperationServer.read NodeManager.read	assigned	assigned	assigned	assigned
Monitor	Download deployed integration flow	IntegrationOperationServer.read NodeManager.read	assigned	assigned	assigned	assigned
Monitor	View deployed security material	IntegrationOperationServer.read NodeManager.read	assigned	assigned	assigned	assigned
Monitor	Add credentials	IntegrationOperationServer.read NodeManager.deploycredentials NodeManager.deploycontent	assigned	n.a.	n.a.	assigned
Monitor	Add known host, keystore, PGP keyring artifacts	IntegrationOperationServer.read NodeManager.deploysecuritycontent NodeManager.deploycontent	n.a.	n.a.	n.a.	assigned

Area	Task	Role	Integration Developer (AuthGroup.IntegrationDeveloper)	Business Expert (AuthGroup.BusinessExpert)	Supporter/ System Developer (AuthGroup.SystemDeveloper)	Tenant Administrator (AuthGroup.Administrator)
Monitor	Edit credentials	IntegrationOperationServer.read NodeManager.deploycredentials NodeManager.readcredentials NodeManager.deploycontent	assigned	n.a.	n.a.	assigned
Monitor	Undeploy credentials	IntegrationOperationServer.read NodeManager.deploycontent NodeManager.deploycredentials	n.a.	n.a.	n.a.	assigned
Monitor	Undeploy known host, keystore, PGP keyring artifacts	IntegrationOperationServer.read NodeManager.deploycontent NodeManager.deploysecuritycontent	n.a.	n.a.	n.a.	assigned

Area	Task	Role	Integration Developer (AuthGroup.IntegrationDeveloper)	Business Expert (AuthGroup.BusinessExpert)	Supporter/ System Developer (AuthGroup.SystemDeveloper)	Tenant Administrator (AuthGroup.Administrator)
Monitor	Download keystore, public/ prviate keyring, known host, .. artifact	IntegrationOperationServer.read NodeManager.read NodeManager.readsecuritycontent	n.a.	n.a.	n.a.	assigned
Monitor	View certificate-to-user mappings	IntegrationOperationServer.read NodeManager.read	assigned	assigned	assigned	assigned
Monitor	Create/edit/ delete certificate-to-user mappings	IntegrationOperationServer.read NodeManager.deploysecuritycontent NodeManager.read	n.a.	n.a.	n.a.	assigned
Monitor	View keystore entries	IntegrationOperationServer.read NodeManager.read	assigned	assigned	assigned	assigned
Monitor	Download public keystore entries	IntegrationOperationServer.read NodeManager.read	assigned	assigned	assigned	assigned

Area	Task	Role	Integration Developer (AuthGroup.IntegrationDeveloper)	Business Expert (AuthGroup.BusinessExpert)	Supporter/ System Developer (AuthGroup.SystemDeveloper)	Tenant Administrator (AuthGroup.Administrator)
Monitor	Add/replace/ delete keystore entries	IntegrationOperationServer.read NodeManager.deploysecuritycontent	n.a.	n.a.	n.a.	assigned
Monitor	View data store entries/variables	IntegrationOperationServer.read ESBDataStore.read	assigned	assigned	assigned	assigned
Monitor	View data store entries - message payload/ variables-content	IntegrationOperationServer.read ESBDataStore.readPayload	n.a.	assigned	n.a.	n.a.
Monitor	Delete data store entries/ variables	IntegrationOperationServer.read ESBDataStore.read ESBDataStore.delete	n.a.	n.a.	n.a.	assigned
Monitor	View payload of stored messages from message store	esbmessagestorage.read	n.a.	assigned	n.a.	n.a.

Area	Task	Role	Integration Developer (AuthGroup.IntegrationDeveloper)	Business Expert (AuthGroup.BusinessExpert)	Supporter/ System Developer (AuthGroup.SystemDeveloper)	Tenant Administrator (AuthGroup.Administrator)
Monitor	View trace configuration	IntegrationOperationServer.read NodeManager.read ConfigurationService.RuntimeBusinessParameterRead	assigned	assigned	assigned	assigned
Monitor	Edit trace configuration (enable/disable trace)	IntegrationOperationServer.read NodeManager.read ConfigurationService.RuntimeBusinessParameterRead ConfigurationService.RuntimeBusinessParameterWrite	assigned	assigned	n.a.	assigned
Monitor	Add/Edit/ undeploy number ranges	IntegrationOperationServer.read NodeManager.deploycontent	assigned	n.a.	n.a.	assigned
Monitor	View number ranges	IntegrationOperationServer.read	assigned	assigned	assigned	assigned

Area	Task	Role	Integration Developer (AuthGroup.IntegrationDeveloper)	Business Expert (AuthGroup.BusinessExpert)	Supporter/ System Developer (AuthGroup.SystemDeveloper)	Tenant Administrator (AuthGroup.Administrator)
Monitor	Retry queues	IntegrationOperation-Server.read ESBDataStore.read ESBDataStore.retry	assigned	n.a.	n.a.	assigned
Monitor	Delete queues	IntegrationOperation-Server.read ESBDataStore.read ESBDataStore.delete	n.a.	n.a.	n.a.	assigned
Monitor	View queues	IntegrationOperation-Server.read ESBDataStore.read	assigned	assigned	assigned	assigned
Monitor	View runtime processing locks	IntegrationOperation-Server.read MessageProcessing-Locks.Read	assigned	n.a.	assigned	assigned
Monitor	Delete runtime processing locks	IntegrationOperation-Server.read MessageProcessing-Locks.Delete	n.a.	n.a.	n.a.	assigned

Area	Task	Role	Integration Developer (AuthGroup.IntegrationDeveloper)	Business Expert (AuthGroup.BusinessExpert)	Supporter/ System Developer (AuthGroup.SystemDeveloper)	Tenant Administrator (AuthGroup.Administrator)
Monitor	Test connectivity	IntegrationOperationServer.read NodeManager.deploycredentials	assigned	n.a.	n.a.	assigned
Monitor	Change log level	IntegrationOperationServer.read ConfigurationService.RuntimeBusinessParameterWrite NodeManager.read	assigned	assigned	n.a.	assigned
Monitor	View audit log entries	IntegrationOperationServer.read AuditLog.Read	n.a.	n.a.	n.a.	assigned
Settings	View/change product profile	WebToolingSettingsProductProfiles.savetenantconfiguration	n.a.	n.a.	n.a.	assigned
Settings	Set ntransport system	WebToolingSettingsProductProfiles.savetenantconfiguration	n.a.	n.a.	n.a.	assigned

4 Provisioning Message Broker

You (tenant admin) can provision message broker to use JMS adapter scenarios only if you have *Enterprise Edition* license.

Context

You can provision the message broker and bind broker to tenant management node and worker node.

Procedure

1. Execute following steps to navigate to your account:
   a. Log into the accounts cockpit.
   b. Choose global account.
   c. Navigate to *Accounts* tab.
   d. Choose a customer account.

      For example, customer account *Display Name* is *CPI PI Productive -abcd1* .
2. Execute following steps to assign tenant admin role:
   a. Navigate to *Subscription* tab.
   b. Choose *provision* application for *hcisvc* provider account.
   c. Navigate to *Roles* tab.
   d. Assign *AuthGroup.Administrator* role to provision application.
3. Execute following steps to provision broker:
   a. Navigate back to your account.
   b. Navigate to *Services* tab.
   c. Choose *Cloud Integration* tile.
   d. Choose *Configure Cloud Integration* link under *Service Configuration*.
   e. Choose *Message Brokers* tab.
   f. You can choose *Provision Broker* in right bottom corner of *Message Brokers* window.

> **Note**
>
> - **Google Chrome**, **Firefox** and **Safari** web browsers support provisioning application.
> - If you want to delete message broker then please raise a ticket.

5 Monitoring (Integration Operations Feature in Eclipse)

The Integration Operations feature provides functions for performing administrative tasks related to runtime clusters and to message monitoring.

This section provides information on the editors and the views of the Integration Operations perspective.

Integration Operations User Interface Areas

Area	Description
Node Explorer	Displays structure of the cluster (with all tenants and nodes).
Message Monitoring	Displays processed messages for tenant selected in Node Explorer.
Deployed Artifacts	Displays deployed artifacts for selected tenant and allows you to create and deploy artifacts.
Data Store Viewer	Displays the content of selected transient data stores available for a tenant.
Properties	Displays • Message processing log (for a message selected in Message Monitoring editor) • Properties for selected nodes • Log information for deployed artifacts (if the Deployed Artifacts editor is opened and an artifact is selected)
Aggregated Data	Displays statistical runtime data for messages (content of this tab depends on whether tenant or node is selected in Node Explorer).
Component Status	Displays status of all components running on the nodes of a cluster.
Tasks	Displays status of current tasks (for example, deployment tasks).
TailLog	Displays newest entries of the log (of a runtime node or a management node).

5.1 Launching the Integration Operations Feature

To launch the Integration Operations, you start the locally installed Eclipse application.

Context

Procedure

1. Install Eclipse and the dependent plug-ins (features) as described in the section referred to below.
2. Specify the connection to the management node as described in the section referred to below.
3. Start Eclipse. Depending on the version of your Integration Operations feature, the following information is displayed in a popup window during Eclipse startup and connection to the management node.

Option	Description
Your Eclipse client is newer than the connected Operations Server.	Your (local) Eclipse client connects to a management node (Operations Server) with an older software version. In that case, not all functions might work as expected.
Your Eclipse client is older than the connected Operations Server.	Your (local) Eclipse client connects to a management node (Operations Server) with an newer software version. In that case, the client should be updated by installing a newer version of the feature. The popup window offers a link to the update function of Eclipse.

Related Information

Installing and Configuring the Tool [page 10]

5.2 Node Explorer (View)

The Node Explorer displays the structure of a cluster.

In particular, the Node Explorer displays the tenant management node and assigned runtime nodes.

At top (below the tenant name), the tenant management node is displayed. The tenant management node has the prefix TM. Below the tenant management node, the runtime nodes are displayed which are assigned to the tenant management node.

Runtime nodes with the following status are displayed:

- LIVE – Node is up and running.
- LAUNCHING – Node has been launched but is not yet live (up and running).
- STOPPING – Node is stopping but not yet shut down.
- FAILED – Node failed during launching operation or due to a crash.
- ERROR – Node is in an error state, for example content deployment request on the node was not successful.
- UPDATING – Node is updating. While in this state, the node is not able to receive further deployment requests.
- INACTIVE – Node is not accessible for external communication.
  Nodes with status INACTIVE are greyed out in the Node Explorer.

> **Note**
>
> When you position the cursor on the tenant name, the following information is displayed in a tooltip:
>
> Version of the integration software
>
> When you position the cursor on a tenant management node or on a runtime node, the following attributes of the node are displayed in a tooltip:
>
> - Name: host name of the node
> - Node Type: IFL or IFL_MAP for runtime nodes, TENANT_MGMT for tenant management nodes
> - Status
> - Last Time Alive
> - Version: Version of integration software
> - VM Size

When you double-click a tenant, the Message Monitoring editor is opened for that tenant.

When you double-click (or click) a runtime node, the content of the currently opened view is updated.

In order to display the properties of a runtime node, position the cursor on the node and choose *Show Properties* in the context menu. The attributes of the runtime node are then displayed in the *Properties* view.

From the Node Explorer you can select the following functions in the context menu:

Cursor Position	Available Functions
Cursor positioned on the tenant	*Show Message Processing Log* Opens the Message Processing Log (MPL). The MPL provides information on the steps during the processing of a message More information: Properties View for Messages - Message Processing Log [page 45]
	Deploy Artifacts ... You can deploy artifacts (integration content or a security artifacts like a keystore). More information: Deploying an Artifact [page 70]

Cursor Position	Available Functions
	Test Outbound Connection ... You can test an outbound connection for the tenant. More information: Testing an Outbound Connection [page 79]
Cursor on tenant management node	*Show Properties* Opens the Properties view for the tenant management node.
Cursor on runtime node	*Show Properties* Opens the Properties view for the runtime node.
	Stop You can stop an individual runtime node.

In the menu bar of the Node Explorer tab, you can select the following functions:

- *Refresh*
- *Open Connection Preferences*
  You can open a dropdown listbox and select one of the previously used management node URLs.

5.3 Trace Configuration Editor

Integration flow developers and/or tenant administrators can now enable tracing for specific integration flows.

If, when you click your tenant in the *Node Explorer* in the *Integration Operations* perspective, the editor displays the *Trace Configuration* tab,please read the following information. Integration flow developers and/or tenant administrators can now enable tracing for specific integration flows.

> **Note**
>
> You must use the latest version of eclipse tooling.

Attributes Displayed in the Trace Configuration Editor

Attribute	Description
Name	Name of the integration flow.
Id	ID of the integration flow.

Attribute	Description
Trace Status	A trace can have the following statuses: • Enabled • Disabled • Expired • Suspended **Note** The status Suspended is displayed in the editor if the SAAS administrator switches off the trace temporarily for operational reasons.
Trace Start Time	The time when tracing started.
Trace Expiry Time	The time when tracing is supposed to end.
User	The user who enables or disables tracing.

Note

- Only *Tenant Admins* and *Integration Developers* can use the *Enable* and *Disable* buttons to enable or disable tracing, respectively.
- You need the *Business Expert* role to view the trace.
- Once enabled, tracing continues for 10 minutes. After 10 minutes, the trace expires, but you can enable it again (without redeploying the integration flow) by choosing *Enable* . If tracing is completed before the expiry time, you can disable tracing by choosing *Disable*.
- In the *Properties* view of an integration flow on the *Trace Configuration* tab, you must not make any changes to the *Trace Level* field because by default the tracing is done at the *Header & Body* level.

5.4 Message Monitoring

The Message Monitoring editor displays messages processed for a tenant according to the filter settings.

To open message monitoring, double-click the tenant in the Node Explorer.

To display messages, specify the following filter criteria:

Filter Options

Filter option ...	Allows you to...
Time	Search for messages processed in a specific time interval. You can select from the following predefined time intervals: • Last minute • Last hour • Last 24 hours • Last 7 days • All Alternatively, you can define your own time interval. To do this, specify Start/End Time and Start/End Date.
Status	Search for messages with a specific status. You can select one of the following values as the status: Completed, Failed, Error, Processing, Retry, All.
Integration Flow	Search for messages associated with a specific integration flow. You can enter a string to filter by integration flow names. Also, the wildcard character `*` can be used. For example: `*m*integrationfl*`
ID	Search for messages associated with a specific MessageGuid or application ID. • MessageGuid Identifies the message uniquely. • Application ID Is set when an `SAP_ApplicationID` header element is specified in the associated integration flow in the Content Modifier step. If (by coincidence) the MessageGuid and the application ID are identical, all messages where either the MessageGuid or the application ID have this value are displayed.

Once you have specified and applied the filter settings, the messages are displayed in a table below the filter settings.

The following information is displayed for each message in separate columns:

Message Attributes

Attribute	Description
Processed At	Ttime when the message processing finished
Status	End-to-end status of message processing
Receiver	Receiver of the message
Processing Time	

Attribute	Description
Integration Flow Name	Display name of the related integration flow (that specifies the details of message processing)
Integration Flow Version	Version of the integration flow
Integration Flow Id	Technical name of the related integration flow
Application Id	Is set when an `SAP_ApplicationID` header element is specified in the associated integration flow in the Content Modifier step.

Selecting a message in the Message Monitoring editor determines the content displayed in the Properties view (message processing log).

Displaying Correlated Messages

To display correlated messages, select the *Include Related Messages* checkbox.

If you have configured a message aggregation use case (using the *Aggregator* step in the integration flow), you have the option of showing the status of the source messages (that are to be aggregated) and of the aggregated message. If you activate this option, the message aggregate is displayed as the first row in the table, and the source messages are listed below.

The message that you are searching for (for example, by entering the corresponding ID in the filter options) is displayed in bold letters.

If a scenario with correlated messages resulted in failed messages, these failed messages are truncated and only the last failed message is shown in the message overview (the number of failed messages is given in brackets next to the status).

Related Information

Message Status [page 36]
Message Processing Log for Specific Integration Flow Steps [page 49]

5.4.1 Message Status

Messages processed on a tenant can have one of the following status.

Message Status

Status	Description
COMPLETED	Message has been delivered to receiver successfully.
PROCESSING	Message is currently being processed.
RETRY	After during message processing an error occurred, a retry has been started automatically.
ERROR	During message processing an error occurred, but no automatic retry has been triggered. However, a manual retry can change the status.
ESCALATED	During message processing an error occurred and no retry has been triggered. For synchronous messages, an error messages is sent to the sender.
FAILED	Message processing failed, message has not been delivered to receiver, and no retries are possible. In other words: FAILED is a final status, message processing ultimatively has failed.

An aggregated message processing log (MPL) can have the following status values:

Status of Aggregated Messages

Status	Description
PROCESSING	Is set as soon as the aggregation process is started and remains as long as the aggregate is still open for further messages. Note that this status can in many situations be shown for quite some time (even days) – depending on the applied aggregation use case.
FAILED	Is set when the aggregated message (after aggregation process has successfully been finished) fails.
RETRY	Is set when the aggregated message (after aggregation process has successfully been finished) runs into an error and sending it is retried.

5.5 Deployed Artifacts

The Deployed Artifacts editor provides an overview of deployed artifacts, for example, integration content and security artifacts.

In particular, the editor displays artifacts deployed on the tenant.

For each artifact, the following information is displayed:

Attributes Displayed in the Deployed Artifacts Editor

Attribute	Description
Name	Nameof the artifact For integration flows the display name is shown.
Version	Version of the artifact
Type	Type of the artifact Possible values: • CREDENTIALS (to specify user credentials for basic authentication) • JAVA_KEYSTORE (to store private and public keys) • SSH KNOWN HOSTS (for known hosts file when using Secure Shell protocol (SFTP)) • PGP_PUBLIC_KEYRING (to store public keys when using the Open Pretty Good Privacy (PGP) standard) • PGP_SECRET_KEYRING (to store public and private keys when using the Open Pretty Good Privacy (PGP) standard)
NodeType	Indicates the node type related to the artifact. Possible values: • IFL - for runtime node • IFLMAP - for runtime node • TENANT_MGMT - for tenant management node **Note** When you deploy a security artifact on a tenant, it will be made available to all runtime nodes (of the tenant). This behavior is made visible by setting the value of the `<NodeType>` attribute to `TENANT_ALL` in the *Deployed Artifacts* editor.

Attribute	Description
Deploy State	Indicates if an artifact has been deployed successfully on a tenant. Possible values: • Stored • Started • Deploying • Deployed • Error Indicates errors during deployment of an artifact. For example, if a wrong passphrase is specified during deployment of a keystore, the keystore will be deployed, but the *Deploy State* gets the ERROR value. The Tasks View also shows this deployment error. **Note** How this state is related to the Synch State attribute is described under: Component Status View [page 52].

When you select an artifact, the logging entries for the deployment of that artifact are displayed in the *Properties* view.

From the *Deployed Artifacts* editor, you perform the following tasks:

- Deploying artifacts
- Undeploying artifacts
- Downloading artifacts
  You **cannot** download any content that has been classified by the content publisher as **configure-only**.

Note

When you select a CREDENTIALS artifact (for User Credentials) you also have the option to edit the artifact (*Edit* button). For more information, see the detailed chapter referred to below under Related Links.

Related Information

Deploying an Artifact [page 70]

5.6 Data Store Viewer

The Data Store Viewer displays the content of selected transient data stores available for a tenant.

A transient data store temporarily stores messages for later processing. An integration flow can write messages to a transient data store or read messages from it.

To open the Data Store viewer, in Node Explorer double-click a tenant and open the *Data Store Viewer* tab.

Users assigned to the following authorization groups can access the *Data Store Viewer*:

- AuthGroup.IntegrationDeveloper
- AuthGroup.Administrator (for the tenant administrator)
- AuthGroup.ReadOnly (read-only)
- AuthGroup.SystemDeveloper (read-only)

The available transient data stores are listed in a table – dependent on the filter criteria.

Filter Options

Filter Option	Shows ...
All Data Stores	All data stores available for the tenant
Only Data Stores with Overdue Entries	All data stores with at least one entry in *overdue* status A message is in status *overdue* in case it has not been processed although the *Wait Time Before Raising Alert* (configured in the corresponding *Write* step which has the transient data store assigned) has been exceeded.

The following attributes are shown for the selected tenant.

List of Data Stores

Attribute	Description
Data Store Name	

Visibility	A transient data store can either be shared across all integration flows deployed on the tenant (global data store) or only be used by one integration flow (local data store). Moreover, a data store can also be used by an *Aggregator* step.
	Depending on the visibility and usage of a transient data store, this column provides the following entry:
	• For a local data store:
	  ○ `integration_flow_name`
	  ○ `integration_flow_name/DataStoreAggregationRepository`
	  In case the transient data store is used by an *Aggregator* step and the aggregation process is in progress
	  ○ `integration_flow_name/DataStoreAggregationCompleted`
	  In case the transient data store is used by an *Aggregator* step and the aggregation process has been completed
	• For a global data store: the string Global

Number of Entries	

Attribute	Description
Number of Overdue Entries	

You can filter by typing any string.

When you position the cursor on an entry and you choose *Copy* in the context menu, you can copy the whole entry (including all table columns).

When you select a data store, the *Properties* view shows the following information (for the selected data store).

Data Store Properties

Attribute	Description
Entry ID	Displays the ID of the data store entry. This ID can be configured in the corresponding *Write* step which has the transient data store assigned.
Overdue	Indicates if the message is overdue (`Yes`) or not (`No`).
Overdue Since	Indicates the time since the message is overdue.
Written at	Provides the time when the message has been received by the integration runtime.
To be Deleted on	Provides the time when the message will be deleted (according to the corresponding *Write* step which has the transient data store assigned).

Up to 500 entries maximum are displayed for a data store by default. In case, there are more than 500 entries, this is indicated, and by clicking a hyperlink you can display the additional entries.

You can filter by typing any string on top of the table.

You can apply the following functions on data store entries. The functions are either accessible in the menu bar of the *Properties* view or by selecting one or multiple entries and choosing the corresponding function from the context menu:

Functions on Data Store Entries

Function ...	Does the following...
Refresh	Refreshes the data store entry list.
Copy Entry ID	Copies one or more entry IDs to the temporary storage. You can select one or multiple entries and then apply this function. In case you selected multiple entries, the IDs of the selected entries are concatenated and delimited by a comma.
Delete Entry	Deletes one or multiple data store entries. You can select one or multiple entries and then apply this function. **Note** To delete an entry, you need permissions as granted for the *AuthGroup.Administrator* authorization group.

Function ...	Does the following...
Download Entry	Downloads an entry to your computer. You can download only one entry at the same time. To download an entry, the role `ESBDataStore.readPayload` (which is part of authorization group `AuthGroup.BusinessExpert`) has to be assigned to your user. **Note** You can only use the download function if you are connected to the TMN. You will get a .zip file. If the downloaded message contains headers, they are contained in the `headers.prop` part of the .zip file. If the downloaded message contains a payload, it is contained in the `body` part of the .zip file.
Get all Entries	Displays all entries. Up to 500 entries maximum are displayed for a data store by default. In case, there are more than 500 entries, this is indicated on top of the table. Using this function, you can display also the additional entries.

5.7 Properties View

The Properties view displays different data, dependend on which entity is selected.

Properties View

Selected Entity	Content of Properties View
Node Explorer: runtime node or the a tenant management node selected	Properties view contains node properties. The Properties view is composed of the following tabs: • Node • Services
Message Monitoring editor: message is selected	Properties view displays the message processing log (MPL) for this message. The MPL provides information on the steps during the processing of a particular message.

5.7.1 Properties View for Nodes

When you select a node, information related to the node are displayed in the Properties view.

5.7.1.1 Node Tab

The following attributes are displayed in the *Node* tab:

Area	Property	Description
General	Application	Specifies the application under which the related tenant is subscribed.
	Node Type	Specifies the node type. Possible values are: • IFLMAP for runtime nodes • TENANT_MGMT for tenant management node
	Profile	Specifies the node profile. Node types and node profiles are related to each other in the following way: • Node type `IFL` relates to node profile `ifl` • Node type `IFLMAP` relates to one of the following node profiles: `iflmap` OR `igwpro*` (where * denotes the SAP Process Orchestration release, see below) • Node type `TENANT_MGMT` relates to node profile `tmn` Node profiles `iflmap` and `igwpro*` are related to the Product Profiles of the associated Web UI in the following way: • Node profile `iflmap` relates to product profile **SAP Cloud Platform Integration** • Node profile `igwpro*` relates to product profile **SAP Process Orchestration** (* denoting the release)
	URL	Specifies URL of the runtime node.

Area	Property	Description
	Version	Indicates the software version of the runtime node (“node repo” version; same attribute as displayed as Version in the Properties View when cursor is positioned on the node) This information is only displayed for nodes in one of the following status: LIVE, REMOVED, STOPPING or UPDATING.
	VM Name	Name of virtual machine
	VM size	Virtual machine size (more information: Starting and Stopping Nodes)
Runtime Info	Last Time Alive	Indicates time when the node has last been in status LIVE
	Status	Indicates status of runtime node that is also displayed in the Node Explorer (tooltip).
	Status Description	Provides optionally more information for the current status.
	Used CPU	
	Used Memory	

> **Note**
>
> The following attributes are only displayed for runtime nodes in status LIVE, ERROR, FAILED, STOPPING: Version, VM Size, Used CPU, and Used Memory.
>
> For nodes in status FAILED, the Version is not displayed in case failed launching of the node has caused this status.

5.7.1.2 Services Tab

The *Services* tab provides information on the service endpoints that are exposed by a `runtime node` selected in the node explorer.

> **Note**
>
> This information allows the integration content developer to check if the expected endpoints are exposed for the configured integration flows.

The tab displays the following attributes for a runtime node:

List of Integration Flow-Specific Endpoints

All integration flows that are processed by this runtime node and expose an endpoint (for Web service connectivity) are listed. For each integration flow, the integration flow-specific provider endpoint is displayed (next to *Endpoint*).

Administrators of remote systems who want to connect the remote system as a consumer to the runtime node need the relevant endpoint information. You have the following options to provide the consumer system administrator with the required information (position the cursor on the endpoint and call the context menu).

Context Menu Options

Option	Allows you to ...	Further Options
Copy Endpoint URL	Copy the endpoint URL that is displayed on the *Services* tab. The endpoint URL can be used to directly call a runtime node. It contains the server name of the **runtime node**.	
Copy WSDL URL	Copy the URL of the WSDL file.	For each of these options, you can choose which kind of WSDL you would like to copy or download. • *Standard* The WSDL includes policies. • *For ABAP Consumer* The WSDL includes those elements that are required to connect an SAP system based on Application Server ABAP. • *Without Policies* The WSDL does not include policies.
Download WSDL	Download the WSDL file.	

For sender channels for which no WSDL has been specified, a **generic WSDL** file can be downloaded. You can use this generic WSDL to configure a WS consumer proxy. For the following adapter types generic WSDLs are available: SOAP (SOAP 1.x), SOAP (SAP RM), and IDoc.

5.7.2 Properties View for Messages - Message Processing Log

When a message is selected in the Message Monitoring editor, the Properties view displays the message processing log (MPL) for this message.

The MPL provides information on the steps during the processing of a particular message.

Related Information

Displaying the Message Processing Log [page 46]
Message Processing Log Properties [page 47]
Message Processing Log for Specific Integration Flow Steps [page 49]

5.7.2.1 Displaying the Message Processing Log

When you select a message in the Message Monitoring editor, the message processing log is displayed in the Properties view.

Context

To display the MPL for a message flow, perform the following steps.

Procedure

1. In Integration Operations perspective, Node Explorer, double-click on the tenant.
2. The Message Monitoring editor is opened.
3. In the Message Monitoring editor click on the row for the message you like to analyse.
4. In the *Properties* view, the MPL is displayed. For information on the displayed attributes, see the detailed section referred to unter Related Links.

   You can download the log to your local hard disk by selecting the *Save* function in the menu bar of the *Properties* view.

Related Information

Message Processing Log Properties [page 47]

5.7.2.2 Message Processing Log Properties

The message processing log displays structured information on the processing of a message.

The following table lists the general properties of the message processing log (MPL):

MPL Properties

Property	Description
ContextName	Integration flow name
CorrelationId	ID that identifies correlated messages Messages can be correlated, for example, when different integration flows on the same tenant communicate with each other. A correlation ID is a base64-encoded ID that is generated in this case by the first integration flow and stored in the message header. As part of the message header, the CorrelationId is then propagated across all related integration flows (that are in charge of processing the correlated messages). Using the OData API, you can read out correlated messages by providing the correlation ID in the OData request.
ComponentType	Runtime component that processed the message
IntermediateError	True if an error occurred (even temporarily) during message processing, or message processing took more than 1 minute.
MessageGuid	Key that identifies the message uniquely in the database
Node	Host name of the node that processed the message
OverallStatus	Specifies the end-to-end status of message processing and corresponds to the *Status* attribute in the Message Monitoring editor. For more information on the statuses, see Message Status [page 36].
ReceiverId	Specifies the name of the receiver as configured in the related integration flow if a SOAP or IDoc endpoint is used.
SenderId	Specifies the name of the receiver as configured in the related integration flow if a SOAP or IDoc endpoint is used.
StartTime	Start of message processing

Property	Description
Status	Specifies the status of message processing with regard to the related part of the MPL data structure. **Note** Note that Status in the MPL data structure refers to the individual message processing step. There is an additional attribute OverallStatus for the whole message processing data structure, which specifies the end-to-end status of message processing. The OverallStatus field corresponds to the Status field in the table view of the MPL.
StopTime	End of message processing
Error	Refers to one or more child data structures (can be empty).

The following properties contain technical information.

Technical MPL Properties

Property	Description
StepID	ID of the related integration flow step

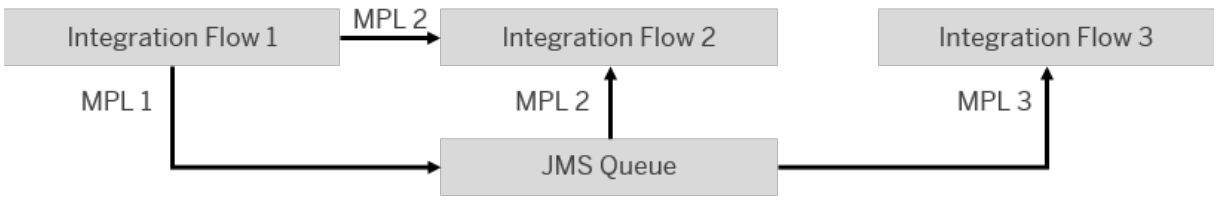
The involved parts of message processing are displayed as separate child structures. On top of the MPL structure, you find proprties that contain metadata of the message as a whole. Below that, you find entries for each relevant integration flow step. As an example, you can find routing conditions (Xpath expressions) evaluated for the message at runtime. In case a mapping is used, the name of the evaluated mapping is shown.

When the corresponding integration flow processes the same sequence of steps multiple times (in a loop), the loops are displayed one after each other on the same level in the MPL structure.

Example for Message Correlation

When different integration flows (on the same tenant) are connected to a JMS queue, from where they store and retrieve messages, the messages (processed by the different integration flows) are correlated by a CorrelationId.

The following figure illustrates that, when three integration flows are used, different message processing logs (MPLs) are written for the overall message processing. However, each MPL uses the same *CorrelationId*, which makes it possible for a user to retrieve all related MPLs for this message processing sequence in one run.



Involved Message Processing Logs (MPLs)

MPL	Description	CorrelationID
MPL 1	Shows that message is received from sender and submitted to JMS queue	XYZ
MPL 2	Shows that • Message is received from JMS queue and submitted to JMS queue • Message is received from JMS queue and a retry is triggered	XYZ
MPL 3	Shows that message is received from JMS queue and submitted to receiver	XYZ

Related Information

Message Processing Log for Specific Integration Flow Steps [page 49]

5.7.2.3 Message Processing Log for Specific Integration Flow Steps

For certain integration flow steps specific message processing log properties are available.

Aggregated Messages

When you have activated the *Include Related Messages* checkbox for an aggregated message in the message monitoring editor, the MPL of the aggregated message will contain all dependent source messages.

The following properties are provided for the aggregated message in the MPL (on the root level of the aggregate):

MPL Properties for Aggregated Messages

Property	Description
AggregateCorrelationId	Identifies the messages which are correlated and are to be aggregated.

Property	Description
AggregateCompletedBy	Indicates how the aggregation has been completed. The following values are possible: • *predicate* The complete set of expected messages has been received • *timeout* The complete set of expected messages has not yet been received and no new messages have been received for the period specified in the Completion Timeout This property is set when the aggregated message has been sent.
AggregateTimedOutAfter	

The following properties (on the root level of the aggregate) are provided for a specific aggregation strategy:

MPL Properties for Aggregated Messages (for Specif Aggregation Strategy)

Property	Description
LastMessageReceived (when *Combine* or *Combine in Sequence* is specified as *Aggregation Algorithm* in the *Aggregator* step)	Indicates if the last message of the aggregate has been received. Possible values: • *true* • *false*
ReceivedSequenceNumbers (when *Combine in Sequence* is specified as *Aggregation Algorithm* in the *Aggregator* step)	Returns the sequence numbers of the received messages (in interval notation). Example: `ReceivedSequenceNumbers =[1,10], [12,20]` indicates that message with sequence number 11 is missing.

5.7.3 Properties View for Deployed Artifacts

When you select a deployed artifact in the Deployed Artifacts editor, information related to the artifact are displayed in the Properties view.

The *Info* tab of the *Properties* view displays the following information related to the selected artifact:

Attributes of Deployed Artifacts

Attribute	Description
Deployed By	Specifies the user who deployed the artifact.
Deployed From	Specifies the IP address of the VM from which the artifact is deployed.
Deployed On	Specifies date and time of the deployment task.
Description	Displays more information on the artifact.
Name	

The following addtional information is only displayed for deployed security artifacts (type `User Credentials`):

Attributes of Deployed Artifacts

Attribute	Description
Credential Type	Specifies the credential type. Possible values: • Default: in case default basic authentication for general SOAP connectivity is chosen • SuccessFactors: in case basic authentication using the SuccessFactors adapter is chosen
User	Specifies the user that is to be authenticated.
Company ID	Specifies the client instance used to connect to the Success-Factors system. This attribute is only displayed for User Credentials artifacts with credential type SuccessFactors.
Server URL	Specifies the URL used to connect to the SuccessFactors system. This attribute is only displayed for User Credentials artifacts with credential type SuccessFactors.

The *Log* tab of the *Properties* view displays logging information on the deployment of the artifact.

5.8 Aggregated Data View

The Aggregated Data view displays statistical data related to message processing.

In the header of the Aggregated Data View tab, the following information is displayed:

- When participant is selected:
  `Statistic for: <selected participant> [Range: Last hour]`
- When node is selected:
  `Statistic for: <participant / selected node> [Range: Last hour]`

In the content area of the Aggregated Data view, the following information is displayed:

- `Integration flow ID`
- `Exchanges Total`: Displays the total number of messages exchanged.
- `Exchanges Failed`: Displays number of failed message exchanged.
- `Mean Processing Time`: Displays the mean processing time for a successful message exchanged.
  In case no messages have been processed successfully, Mean Processing Time = 0 is displayed.
- `CXF(in/out) Invocations`: Displays total number of CXF invocations.
- `CXF(in/out) Failures`: Displays number of failed CXF invocations.
- `CXF(in/out) Mean Processing Time`: Displays mean processing time for a successful CXF invocation.

For the CXF invocations, the suffix in indicates inbound Web service calls, the suffix out indicates outbound Web service calls.

> **Note**
>
> By default, the filter is set in a way that information is displayed for nodes that have been active during the last hour. To display information related to nodes that have been active earlier, change the filter settings accordingly.
>
> To change the filter settings, click the triangle icon in the menu of the view.

5.9 Component Status View

The Component Status View shows the current status of all components running on the nodes of a cluster.

To display the status of all components of a node, select the node in the Node Explorer and go to the Component Status view.

For each component, the following information is displayed in separate columns:

Attribute	Description
Name	Name of the component For more information on the different components, see separate topic. In´case the component is an integration flow, the display name of the integration flow is shown.
Version	Version of the component

Attribute	Description
Type	Component type The following kinds of components are detected in the Component Status View: • Integration Flow Artifact that specifies the details of how a message is processed by a node. • Value Mapping Artifact that specifies transformations during message processing. • Keystore Artifact that contains key pairs for secure connectivity. • SAP Cloud Platform Integration subsystem. Indicates a service running on the node which is responsible for a specific aspect of message processing. To give an example, there is a subsystem for Web service connectivity (WS Connectivity). **Note** **Note** An *artifact* indicates an element that can be deployed on a tenant and be made available to all runtime nodes (assigned to the tenant).

Attribute	Description
Runtime status	Indicates if a component is operational on the runtime node. **Note** The runtime status is the same attribute as displayed as *Status* in the Aggregated Data View. For components with status error, you can display error details by positioning the cursor on the corresponding component (table row) in the Component Status view and selecting *Show Error Details* in the context menu. Error details can be displayed for subsystems as well as for deployable artifacts. **Note** For an *operations agent* component in status error, the Error Details show which related components are erroneous. With regard to deployable content such as integration flows and keystores, for example, note the difference between the *Synch State* (displayed in the *Deployed Artifacts* editor) and the *Runtime Status* (displayed in the *Component Status View*). To illustrate the difference, there are the following examples: Possible error causes depend on the component. For examples of error causes, see table below. More information: Runtime Status [page 56]

Attribute	Description
Synch Status	Indicates if an artifact which has been deployed on a tenant has already been replicated and made available to the assigned runtime node. **Note** The Deploy State displayed in the Deployed Artifacts editor indicates if an artifact has been deployed successfully on a tenant. After the deployment, the tenant management node and the assigned runtime nodes are being synchronized. After this synchronization process, the artifact is been replicated and made available for the runtime nodes. The result of this process is indicated with the Synch Status attribute in the Component Status view. This attribute can have the following values: • `synchronized` Component on selected runtime node is identical with component on tenant management node. • `to be updated` Component on selected node has different ID than component on tenant management node, however, both components have the same name. Therefore, this state implies that the component is to be considered as to be updated. • `to be removed` Component on selected runtime node doesn't exist on tenant management node at all (by component name). Therefore, this state implies that the component is to be considered as to be deleted. • `unknown` Component on selected runtime node cannot be related to a component on the tenant management node (by name). That means, it is not known to the system and no ID has been set for the component. A possible reason for this situation is that this component has been made available for the runtime node by any manual method but not by the standard deployment process. **Note** This status is only displayed in case the component type is an artifact (integration flow or keystore). • `not applicable`

Attribute	Description
	This Synch Status is not displayed for *subsystem* component types, because, other than an integration flow or a keystore, a subsystem cannot be deployed and therefore, a synch status has no meaning.

> **Note**
>
> Note the following difference between Aggregated Data view and Component Status view with regard to deployed content:
>
> The Aggregated Data view shows the operative state of the content (for example, if an integration flow runs without errors). The Aggregated Data view shows only the status for integration flow content.
>
> The Component Status view shows if content has reached the node during deployment as well as the actual runtime status of the deployed bundle.

Related Information

Runtime Status [page 56]
Component Monitors [page 57]

5.9.1 Runtime Status

The runtime status indicates if a component is operational on the runtime node.

The following values are possible:

- `installed`
  For a deployable component, this status means: the component has been deployed but not started yet.
- `launching`
  Component is currently being started.
  For a deployable component, this status means: the component has been deployed and is currently being started.
- `started`
  Component is started on selected runtime node.
- `error`
- `stopping`
  Component is currently being stopped.

5.9.2 Component Monitors

The Component Status view provides an overview of the status of the components on the cluster nodes. This section provides an overview of the available components as well as information on the recommended steps in case a component is in state error.

Overview of Components and Recommended Steps in Case of Errors

Component	Component Type	Description	Steps to Resolve Erroneous Runtime State
Camel AHC	Subsystem	Checks the availability of the Camel HTTP component. This monitoring subsystem runs on runtime nodes.	Open a ticket on **LOD-HCI-PI-RT**.
Camel-core	Subsystem	Checks the availability of core functions of Camel (like the pipeline engine and XSLT mapping engine). This monitoring subsystem runs on runtime nodes only.	Open a ticket on **LOD-HCI-PI-RT**.
Camel FTP	Subsystem	Checks the availability of the Camel SFTP component. This monitoring subsystem runs on runtime nodes.	Open a ticket on **LOD-HCI-PI-RT**.
Camel Idoc Soap	Subsystem	Checks the availability of the IDoc Soap component. This monitoring subsystem runs on runtime nodes.	Open a ticket on **LOD-HCI-PI-RT**.
Camel Mail	Subsystem	Checks the availability of the Camel mail component. This monitoring subsystem runs on runtime nodes.	Open a ticket on **LOD-HCI-PI-RT**.
Camel Scheduler	Subsystem	Provides Camel scheduling functions.	Open a ticket on **LOD-HCI-PI-RT**.

Component	Component Type	Description	Steps to Resolve Erroneous Runtime State
Camel Security	Subsystem	Checks the availability of Camel Security components, like digital signature und encryption based on PKCS#7, PGP, and digital signature based on XML Digital Signature. This monitoring subsystem runs on runtime nodes only.	Open a ticket on **LOD-HCI-PI-RT**.
Camel WS Connectivity	Subsystem	Checks the availability of the Camel Web service componant. This subsystem runs on tenant management nodes and runtime nodes.	Open a ticket on **LOD-HCI-PI-RT**.
Catalog	Subsystem	Displays the catalog tab in Web UI.	Open a ticket on **LOD-HCI-PI-CST**.
CXF-endpoint-<node type>-<virtual server>	Subsystem	Identifies a component that monitors automatic connectivity tests. This component is only displayed for a tenant management node in case there are active runtime nodes assigned to the tenant management node. Using this component, you can monitor if runtime nodes (assigned to the tenant management node) can be reached by external calls. More information: Monitoring External Reachability of Runtime Nodes [page 66]	Check out the error message. **Note** A possible error cause is that no suitable keystore is deployed. The connectivity test works that way that the tenant management node calls a runtime node (assigned to the tenant management node) via the load balancer through SSL. To enable SSL-based connectivity, a keystore has to be deployed on the tenant that fulfils the following requirements: • It contains a root certificate that is accepted by the load balancer. • It contains a client certificate (public/private key pair) that is accepted by the load balancer. In case there is a problem with the keystore, first correct this by deploying a keystore with the suitable content. If the component remains still in an errorneous state, open a ticket on **LOD-HCI-PI-RT**.

Component	Component Type	Description	Steps to Resolve Erroneous Runtime State
Generation and Build	Subsystem	This subsystem indicates the availability of the integration flow deployment service. This monitoring subsystem is available on tenant management nodes only.	Open a ticket as per below guidelines. If the error indicates that any of the bundles in erroneous state, open a ticket on **LOD-HCI-PI-GB**. If the error indicates that any of the services starts wth `<com.sap.it.gb>` in erroneous state, open a ticket on **LOD-HCI-PI-GB**. If the error indicates that the `<com.sap.ifl.gnb.generator>` service in erroneous state, open a ticket on **LOD-HCI-PI-ET-IFL**. If the error indicates that the one of the services `<com.sap.it.ictools.mapping.generation.integration>` or `<com.sap.volante.generator.intgn>` is in erroneous state, open a ticket on **LOD-HCI-PI-RT-MAP**. For all other erroneous states, open a ticket on **LOD-HCI-PI-GB**.
<integration flow name>	Integration Flow	Appears when an integration flow is deployed on the runtime node. The monitor checks whether the integration flow was starting correctly. This monitoring subsystem is runs on runtime nodes only.	Check out the error message first. Possible error causes can be: • The required security artifacts are missing (for example, a required keystore or basic authentication artifact is not deployed). If this is the error cause, deploy the required artifact and restart the integration flow (by selecting the integration flow in the Component Status view and choosing *Restart*). • The integration flow design is inconsistent. As an example of an inconsistently designed integration flow, assume the integration flow specifies communication with an SFTP server via Secure Shell protocol (SFTP) and the polling of data from the SFTP server fails (for example, in case ssh key pairs are wrongly configured or not deployed). If this is the error cause, correct the integration flow, re-deploy it and restart it (by selecting the integration flow in the Component Status view and choosing *Restart*).

Component	Component Type	Description	Steps to Resolve Erroneous Runtime State
Message Store	Subsystem	This monitor checks the availability of the Data Store component. This subsystem runs on tenant management nodes and runtime nodes.	Open a ticket on **LOD-HCI-PI-RT**.
Message Service (JMS)	Subsystem	Checks the availability of the Message Service for the inter node communication. This subsystem runs on all node types. In case the status of this component is not `started`, the message service is not available. In that case, in the Integration Operations user interface tenant management nodes and runtime nodes are not displayed and cannot be started or stopped either.	Open a ticket on **LOD-HCI-PI-RT-MSG**.
OData Adapter	Subsystem	Monitors the adapter that provides services to connect to the OData provider. This is done by checking whether particular bundles are started and are running without errors. This subsystem runs on runtime nodes only.	Open a ticket on **LOD-HCI-PI-GB**.

Component	Component Type	Description	Steps to Resolve Erroneous Runtime State
Operations Agent	Subsystem	. This monitor checks the availability of basic monitoring services on the runtime node. This is done by checking whether particular bundles are started and are running without errors and particular services components are active. These services cover, for example, Camel and CXF collectors, error detection and MPL storage. This subsystem runs on runtime nodes only.	Open a ticket on **LOD-HCI-PI-OP-SRV**.
Operations Server	Subsystem	This monitor checks the availability of basic monitoring services on the management nodes. This subsystem runs on tenant management nodes only. It covers subsystems for logging and alerting as well as servlet and command processing to enable user interface clients to connect to this node.	Open a ticket on **LOD-HCI-PI-OP-SRV**.
Operations Server Connectivity	Subsystem	This monitor checks if the tenant management node can be reached by external calls. More information: Monitoring External Reachability of Tenant Management Nodes [page 65]	Check out the error message in the error details. Open a ticket on **LOD-HCI-PI-OP-SRV**.

Component	Component Type	Description	Steps to Resolve Erroneous Runtime State
Persistence	Subsystem	This monitor checks the availability of the database and the corresponding persistence service layer. This subsystem runs on tenant management nodes and on runtime nodes. **Note** This component on a regulary basis writes the following information into the database: CPU and memory used by the node that is associated with the node.	
PGP	Subsystem	Monitors the subsystem responsible to access the content of PGP keyrings. This monitoring subsystem is runs on runtime nodes only	Open a ticket on **LOD-HCI-PI-RT**.
Scheduler Subsystem	Subsystem	Monitors the subsystem responsible for scheduling (operations jobs and integration flow routes). This subsystem runs on tenant management nodes and runtime nodes.	Open a ticket on **LOD-HCI-PI-GB**.
Scheduler	Subsystem	Subsystem related to scheduling re-occuring system tasks. This subsystem runs on runtime nodes.	Open a ticket on **LOD-HCI-PI-RT**.
Security	Subsystem	This monitor checks the availability of basic security services like encryption and singing engines. This subsystem runs on tenant management nodes and runtime nodes.	Open a ticket on **LOD-HCI-PI-RT**.

Component	Component Type	Description	Steps to Resolve Erroneous Runtime State
SuccessFactors Adapter	Subsystem	Identifies a sub system that provides services to monitor the SuccessFactors adapter. The SuccessFactors adapter provides services to connect to the Success Factors system. This is done by checking whether particular bundles are started and are running without errors. This subsystem runs on runtime nodes only.	Open a ticket on **LOD-HCI-PI-GB**.
Tenant Configuration Sync Monitor	Subsystem	This subsystem runs on the tenant management nodes and shows if the actual state of the runtime nodes of the cluster corresponds to the settings specified in the Tenant Configuration. To ensure that the up-to-date status of the cluster is displayed, refresh the view.	
Transient Data Store	Subsystem	This monitor checks the availability of the Data Store component. This subsystem runs on tenant management nodes and runtime nodes.	Open a ticket on **LOD-HCI-PI-RT**.

Component	Component Type	Description	Steps to Resolve Erroneous Runtime State
<name of keystore>	Keystore	Identifies a deployed keystore. This component is displayed for tenant management nodes and runtime nodes.	Check out the error message. **Note** A possible reason for the runtime status of a key store to change to `error` is, for example, that the key store does not contain any private keys or certificates.When the key store is deployed with a correct password, this is still a valid key store with regard to deployment, and therefore the Deploy State doesn't change to `error`. However, this key store cannot be used at runtime, therefore, the runtime status changes to `error`. If the error message points you to this kind of error cause, check the consistency of the keystore.
<name of keystore>	PGP Public Keyring	Appears if a PGP public keyring (contains public keys for the usage of PGP) is deployed. This monitoring subsystem is runs on runtime nodes only.	Check out the error message. If you cannot solve the problem, open a ticket on **LOD-HCI-PI-RT**
<name of keystore>	PGP Secret Keyring	Identifies a deployed PGP secret keyring (contains public/private key pairs for the usage of PGP). This monitoring subsystem is runs on runtime nodes only.	Check out the error message. If you cannot solve the problem, open a ticket on **LOD-HCI-PI-RT**
Value Mapping	Value Mapping	Identifies a value mapping (deployable content).	
Web Tooling	Subsystem	Identifies the integration flow in the web tool.	Open a ticket on **LOD-HCI-PI-WT**
Workspace	Subsystem	Displays the design tab in Web UI.	Open a ticket on **LOD-HCI-PI-CST**.

Component	Component Type	Description	Steps to Resolve Erroneous Runtime State
WS Connectivity	Subsystem	Checks the availability of the basic WS Connectivity services used by Camel components. This subsystem runs on tenant management nodes and runtime nodes.	Open a ticket on **LOD-HCI-PI-RT**.

With regard to artifacts like integration flows, regard the following:

Artifacts are deployed from a tenant management node to the related runtime nodes. At runtime, it has to be made sure that the version of the artifact on the runtime node is the same like the version on the tenant management node. The Component Status view helps you to analyze issues related to that.

Components of type *Integration Flow* or *Value Mapping* can be restarted. To do this, select the component and choose *Restart* (next to the table).

5.9.3 Monitoring External Reachability of Tenant Management Nodes

Using the Component Status view, you can monitor if the tenant management node can be reached by external calls.

For this purpose the tenant management node calls itself regularly every 30 seconds through the load balancer.

The call simulates the *Test Connection* function in the Integration Operations cockpit. The corresponding monitor is called *Operations Server Connectivity* in the Component Status view (for the selected tenant management node).

The component displays an error status if the tenant management node is not reachable via the load balancer for any reason. In this case, the detailed error message displays either the exception (text) or the HTTP return code and text of the last attempt.

> **Note**
>
> You can display the error message in the Component Status view by positioning the cursor on the component and selecting *Show Error Details* in the context menu.

5.9.4 Monitoring External Reachability of Runtime Nodes

Using the Component Status view, you can monitor if runtime nodes (assigned to the tenant management node) can be reached by external calls. For this purpose, an external SSL call of a runtime node is simulated and monitored using a specific component in the Component Status view.

This monitor displays the basic technical SSL connectivity of a runtime node. To simulate an external call, the following call is performed every 30 seconds on a regular basis:

For sakes of completeness, note that always a tenant keystore (not depicted in the figure) needs to be available to enable the system to do an additional outbound communication step that is required for technical purposes: The basic technical connectivity of a cluster is checked on a regular basis, as soon as the cluster is active. For this purpose, every 30 seconds the tenant management node sends an HTTPS request to an assigned runtime node via the load balancer. This simulates an external call to the runtime node. To enable this communication, a keystore needs to be deployed on the tenant, containing a valid client certificate that is accepted by the load balancer as well as the root certificate of the same. If this keystore is not available or contains an invalid certificate, the cluster will raise an error. The keystore and required certificate are provisioned by SAP together with the tenant.

As monitor for this connectivity test, the following component is displayed in the Component Status view (for the selected tenant management node) as soon as a runtime node has been started:

`CXF-endpoint-<node type>-<virtual server>` (for example: `CXF-endpoint-IFLMAP-intaas`)

The component displays an error status in one of the following cases:

- The required keystores for the SSL connection are not deployed (or not consistent).
  The keystore must contain a a valid client certificate that is accepted by the load balancer as well as the root certificate of the same.
- The runtime node (for which connectivity is to be monitored) does not exist.

5.10 Tail Log View

This view displays the newest entries of the log of a selected node (“tail of the log”). Use this view to search for information in case a node is in an erroneous state.

You can display the tail log of runtime nodes and of tenant management nodes – depending on what is selected in the Node Explorer:

- When you select `cluster` or a participant in the Node Explorer, the tail log for the tenant management node is displayed.
- • When you select a runtime node or a tenant management node in the Node Explorer, the tail log for the node is displayed.

To display the tail log of a node, perform the following steps:

1. Select the node in the Node Explorer and go to the *TailLog View* tab.
2. Specify the size of the tail log (in kB) that should be displayed.
3. Using the *Search* window, you can specify specific strings that are to be highlighted in the tail log.
   You can navigate (upward or downward) to the next hit.

You can refresh the view.

> **Note**
>
> Recommendation: Check the tail log of a runtime node in case you detect problems with this runtime node in the Component Status View.

> **Note**
>
> Recommendation: Check the tail log of a tenant management node in case of general problems with runtime nodes (for example, in case there are problems with starting (launching) runtime nodes or content deployment).

5.11 Tasks View

This view shows the status of the recent tasks performed, such as deploying content or launching a runtime node.

The following attributes are displayed for each task:

- `Task`
  Provides the name of the task.
- `Status`
  The following status values are possible:
  - `new`
    Task is waiting.
  - `Running`
    Task is in progress.
  - `Success`
    Task has been performed successfully.
  - `error`
    Task has been stopped with an error, a retry is possible.
  - `failed`
    Task has been stopped with an error, no retry is possible.
  - `scheduled`
    Task has been scheduled but not been started yet.
- `Start Time`
  Indicates the start time of the task.
  In case a recurring task (job) is displayed, Start Time indicates the time the job has been started first.
- `TenantID`
  Identifies the tenant related to the task.
- `User`
  Displays the user ID that you have used to log on to the Operation Server.
- `TaskID`
  Provides the ID of the task.

The following tasks are displayed:

- Launch node
- Update software
- Shutdown node
- Cleanup
- Node discovery
- Node failover
- Node health check
- Node self update
- Send node snapshot
- Deploy content
- Undeploy content
- Artifact migration
  Is displayed when a cluster with deployed security content is updated.
- Generate and Build Project [<Bundle name from Manifest.MF file is shown in this bracket>] - Task related to single project deployment
- Generate and Build Project [Multiple Projects] - Task related to multiple project deployment

> **Note**
>
> By default, the following attributes are displayed for a task: Task (name), Status and Time.
>
> Using the dropdown list in the toolbar of the view (triangle icon), you can show or hide the Task ID and User column.
>
> You can also refresh the view in order to display the latest state.
>
> Using the dropdown list in the toolbar of the view (triangle icon), you can choose between horizontal and vertical layout for the Tasks View (*Layout* *Horizontal/Vertical*).

When you select the tenant, the tenant management node or a runtime node in the Node Explorer, the *Tasks View* shows the tasks related to the corresponding participant.

When you select the tenant in the *Node Explorer*, the *Tasks View* shows the tasks that are related to the tenant.

When you select a task, a detailed log is displayed under *Task Trace*. The log in the *Task Trace* is available for 24 hrs from the time the task was triggered.

Below you can display more information on each selected task (in the *Task Trace*).

5.12 Executing Restart

You can use the restart feature to start new nodes and stop old nodes.

Prerequisites

- You have set up the cluster.

Context

The restart takes 15 minutes to complete per node. You can restart for the following reasons:

- A new platform runtime version is available.
- A runtime version is pinned.

> **Note**
>
> You must restart the CMN after pinning.Then, execute step 1.

- Using the hot restart feature to recover the node does not work.
- Started node has instability in the bundles.

Procedure

1. To restart a super cluster (all tenants in CMN), proceed as follows:
   - In the *Node Explorer*, execute the following steps:
     1. Position the cursor on the cluster.
     2. In the context menu, choose *Restart*.
2. To restart a node, proceed as follows:
   - In the *Node Explorer*, execute the following steps:
     1. Position the cursor on the node.
     2. In the context menu, choose *Restart*.
3. To restart a tenant cluster, proceed as follows:
   - In the *Node Explorer*, execute the following steps:
     1. Position the cursor on the tenant.
     2. In the context menu, choose *Restart*.

5.13 Deploying an Artifact

You can deploy different types of artifacts (such as integration content or security-relevant artifacts), each serving a different purpose. To make an artifact available for the runtime, you have to deploy it on the relevant runtime node of a given tenant.

Prerequisites

In the Node Explorer, you have positioned the cursor on the tenant and chosen *Deploy Artifact ...* in the context menu.

Context

Procedure

1. Select the artifact type.

Option	Description
Keystore	This artifact contains the public and private key used for certificate-based authentication when sending a message from a tenant to a receiver (outbound message processing). It has to be deployed on the corresponding tenant to enable the tenant to communicate based on public key technology. The keystore artifact is used to deploy both keystores for SSL and SSH transport security. **Note** As of node assembly version 2.29.* and higher, you use the *Keystore* editor in Web UI to deploy this artifact type.
Known Host (SSH)	Specifies the known_hosts file used when configuring secure connectivity based on the SSH File Transfer Protocol (SFTP). It contains the public keys and addresses of the "trusted" SFTP servers. The client checks if the server is a trusted participant by evaluating a known_hosts file on the client side: If the server's public key is listed there, the identity of the server is confirmed.
PGP Public Keyring	This artifact contains the public key that enables the tenant to encrypt or verify messages using the Open Pretty Good Privacy (PGP) standard.
PGP Secret Keyring	This artifact contains public and private key pairs for the usage of Open Pretty Good Privacy (PGP). The private key enables the tenant to decrypt or sign messages.

Option	Description
OAuth2	This artifact contains the OAuth login URL to connect to the service provider. The client ID and client secret verifies the identity of the client.
User Credentials	Specifies user, password (and, depending on the related connectivity option, other attributes) for basic authentication.

> **Note**
>
> If you have missed an attribute or you have entered inconsistent passwords, you cannot proceed with the wizard.

The node type is typically predetermined according to the node type that has been specified for the cluster.

2. Choose *Next*.

Next Steps

To undeploy an artifact, select the artifact in the *Deployed Artifacts* editor and choose *Undeploy*. Confirm with *Yes* in the next window.

> **Note**
>
> You can also deploy content from the *Deployed Artifacts* editor.

Related Information

Deploying and Editing a User Credentials Artifact [page 76]
Deploying a Keystore [page 72]
Deploying a Known Hosts Artifact [page 73]
Deploying a PGP Public Keyring [page 73]
Deploying a PGP Secret Keyring [page 74]
Deploying an OAuth2 Authentication Artifact [page 75]

5.13.1 Deploying a Keystore

This artifact contains the public and private key used for certificate-based authentication when sending a message from a tenant to a receiver

Prerequisites

In the Node Explorer you have selected a tenant, in the context menu you have selected *Deploy Artifacts...*, and then as Artifact Type you have specified *Keystore*.

You have created a keystore file and stored it on your computer.

Context

> ⚠ Caution
>
> This feature is only available if you have a node assembly version up to 2.28.*.
>
> As of node assembly version 2.29.* and higher, you use the *Keystore* editor in Cloud Integration Web UI to deploy this artifact type.
>
> We recommend that you update your cluster to node assembly version 2.28.* (or higher, if available) to use the newkeystore management feature of the Web UI.

Procedure

1. Browse for the keystore file on your computer.
2. Enter the password of the keystore.

   This password has been defined when the keystore has been created.

   When you enter the password, it is checked if the password is identical to the one specified when the keystore has been created. In case the entered password differs from the original one, an error message is displayed. Also in case the keystore is defect and, therefore, cannot be deployed, an error is displayed. Note that only keystores with the format jks or jceks can be deployed.

   Although it is possible to have a keystore without password protection, it is currently (by intention) not possible to deploy an unprotected keystore via the Integration Operations perspective.
3. Select *Finish*.

5.13.2 Deploying a Known Hosts Artifact

This artifact type specifies the known hosts file used when configuring secure connectivity based on SSH File Transfer Protocol (SFTP). It contains the public keys and addresses of the trusted SFTP servers.

Prerequisites

In the Node Explorer you have selected a tenant, in the context menu you have selected *Deploy Artifacts...*, and you have specified *Known Hosts (SSH)*as the Artifact Type.

You have created a known hosts file and stored it on your computer.

Context

Procedure

1. Browse to the known hosts file on your computer.
2. Choose *Finish*.

5.13.3 Deploying a PGP Public Keyring

This artifact contains the public key that enables the tenant to encrypt or verify messages using the PGP standard.

Prerequisites

In the Node Explorer you have selected a tenant, in the context menu you have selected *Deploy Artifacts...*, and you have specified *PGP Public Keyring* as the Artifact Type.

You have created a PGP public keyring file and stored it on your computer.

Context

Procedure

1. Browse to the PGP public keyring file on your computer.
2. Choose *Finish*.

Related Information

How OpenPGP Works [page 234]

5.13.4 Deploying a PGP Secret Keyring

This artifact contains the public and private key pair for the usage of Open Pretty Good Privacy (PGP). The private key enables the tenant to decrypt or sign messages.

Prerequisites

In the Node Explorer you have selected a tenant, in the context menu you have selected *Deploy Artifacts...*, and you have specified *PGP Secret Keyring* as the Artifact Type.

You have created a PGP secret keyring file and stored it on your computer.

Context

Procedure

1. Browse to the PGP secret keyring file on your computer.
2. Enter the password of the PGP secret keyring.
   This password was defined when the PGP secret keyring was created.

   If the entered password does not match the original one, you cannot deploy the artifact.
3. Choose *Finish*.

Related Information

How OpenPGP Works [page 234]

5.13.5 Deploying an OAuth2 Authentication Artifact

Many web servers can use OAuth 2.0 for authorization purposes. If you want to connect to a system that uses OAuth 2.0 authentication, you need to deploy an OAuth2 Credentials artifact using the following procedure.

Context

> **Note**
>
> You can deploy OAuth 2.0 artifact using the *Deploy Credentials* wizard. For information on deploying credentials, see

Procedure

1. Enter values in fields based on the description given in the table:

Attribute	Description
Name	Name of the artifact that you want to deploy on the tenant
Description	Description of the artifact name you are deploying on the tenant
Authentication URL	URL of the OAuth authorization server that issues the access token
Client ID	ID of the client that you are connecting to
Client Secret	Secret key of the client that you are connecting to OAuth uses a multiple step authentication pattern: Client credentials (*Client ID* and *Client Secret*, as specified in the artifact) are used by the client application to initially request an access token. The access token is then used to authorize the client (for as long as the token is valid) to access the server's resources (for example, the resources that are used in the associated integration flow). In many OAuth scenarios, the access token is issued (or generated) by an authorization server.
Scope	Access rights that you are requesting from the service provider

2. Choose *Finish*.

5.13.6 Deploying and Editing a User Credentials Artifact

To enable a runtime node to connect as a client to a receiver system using basic authentication or username token authentication, you have to specify the required attributes (for example, user name and password).

Prerequisites

In the Node Explorer you have selected a tenant, in the context menu you have selected *Deploy Artifacts...*, and you have specified *User Credentials* as the Artifact Type.

Context

You can specify basic authentication either for a connected SuccessFactors system (through the SuccessFactors adapter) or for general SOAP connectivity. To enable basic authentication for a runtime node, you specify the user credentials and deploy the attributes (in the form of a User Credentials security artifact) on the corresponding tenant.

This artifact type is also referred to as *CREDENTIALS* in the Deployed Artifacts editor. In other words: The attributes specified with a User Credentials artifact are summarized as *credentials*.

Procedure

1. Select one of the following values as `Type`:

Option	Description
Default	For general SOAP connectivity
SuccessFactors	For connectivity based on the SuccessFactors adapter

2. Specify the other attributes on this wizard page.

   Specify the following attributes:

   `Name`: Provide a name for the artifact (also referred as *credentials name*).

   `Description`: You have the option of providing a more elaborate description of the artifact.

   `User`: Specify the user that calls the receiver system.

   `Password`: Specify the password against which the user has to be authenticated.

3. Finalize the deployment of the artifact. The following steps depend on the chosen credential type (either Default or SuccessFactors).

Option	Description
You have selected `Default`.	Choose *Finish*.
You have selected `SuccessFactors`.	Choose *Next* and specify additional attributes on the next wizard page.

4. If you have selected SuccessFactors as Type: Choose *Next* and specify the following attributes on the next wizard page.

   Specify the following attributes:

Attribute	Description
Server URL	Specify the URL used to connect to the SuccessFactors system. Example: `https://salesdemo4.successfactors.com:443/sfapi/v1/soap`
Company ID	Specify the client instance used to connect to the SuccessFactors system. Example: `myCompany25`

5. Choose *Finish*.

Next Steps

> **Note**
>
> When you have re-deployed this artifact, you also have to re-deploy and restart all integration flows that use this artifact.

Alternatively, you can deploy User Credentials artifacts in the following way:

- Start the *Deploy Artifacts* wizard by choosing *Deploy* in the *Deployed Artifacts* editor (for a selected tenant).
  In the *Deployed Artifacts* editor you get an overview of all deployed artifacts.

  > **Note**
  >
  > User Credentials artifacts are indicated in the Deployed Artifacts editor by the CREDENTIALS type.

You can also use this wizard to edit an existing User Credentials artifact (from the *Deployed Artifacts* editor).

> **Note**
>
> Note the following when editing an existing User Credentials artifact:
>
> - Specifying a password is not mandatory. However, note that each entry you make when editing the artifact overwrites existing passwords defined for the artifact prior to using this editor. This is also the case when you provide an empty string for the password.

- If a password has already been defined for the artifact prior to using this editor, you need to re-enter this password. If you do not enter any password, the original password will be overwritten (by an empty string).

Related Information

Properties View for Deployed Artifacts [page 50]

5.13.7 Deploying a Secure Parameter Artifact

Use the secure parameter artifact to deploy confidential data for custom adapters.

Prerequisites

In the Node Explorer you have selected a tenant, in the context menu you have selected *Deploy Artifacts*, and you have specified *Secure Parameter* as the artifact type.

Context

You can store secure data like passwords in a secure store and use an alias name to access this data in an integration flow. The secure parameter artifact contains this alias name and confidential data.

You can deploy the artifact on the tenant by following the procedure below.

Procedure

1. Specify the attributes on the wizard page.

Option	Description
Name	Provide a name for the artifact. The artifact name is used as an alias for the confidential data assigned by this parameter.
Description	Provide a more detailed description of the artifact.
Secure Parameter	Enter the confidential value of the attribute.
Repeat Secure Parameter	Repeat the confidential value of the attribute.

2. Choose *Finish*.

> **Note**
>
> If correct alias is configured in the integration flow, then the runtime framework passes the secured artifact value to the component at runtime from the secure store.

5.14 Testing an Outbound Connection

You can test an outbound connection (for a tenant calling a receiver system).

Context

You can check SSL, SSH, and SMTP connections.

Procedure

1. In the Node Explorer, position the cursor on the tenant.
2. In the context menu choose *Test Outbound Connection ...*.
3. Select the type of connection (*SSL Connection*, *SSH Connection*, or *SMTP Connection*).
4. Choose *Next*.
5. Depending on the chosen connection type, specify the following settings for the connection:

Option	Description
For SSL connections	Specify the settings as explained under SSL Outbound Connection Test [page 80].
For SSH connections	Specify the settings as explained under SSH Outbound Connection Test [page 82].
For SMTP connections (when a receiver Mail adapter has been configured)	Specify the settings as explained under SMTP Outbound Connection Test [page 84].

6. Choose *Run*.

Related Information

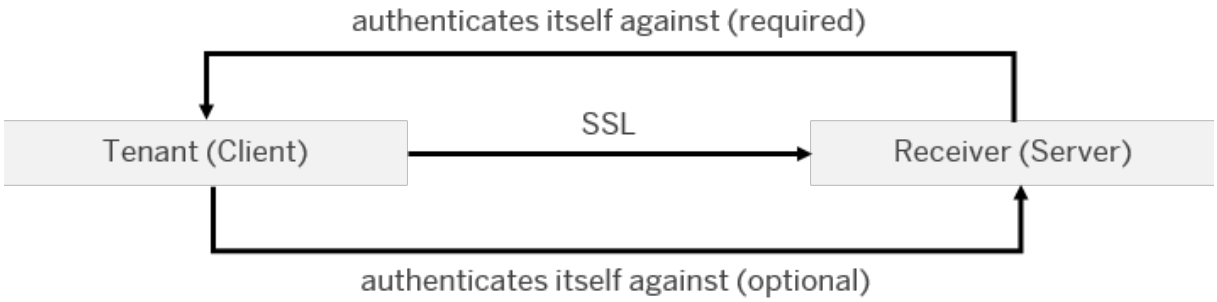
SSL Outbound Connection Test [page 80]
SSH Outbound Connection Test [page 82]

5.14.1 SSL Outbound Connection Test

When you have selected the SSL connection type, the test tool checks the following:

- If the receiver (host) is reachable for the tenant
- If the keystore is deployed correctly and contains those keys that are required for the specified authentication method during SSL handshake

The following figure illustrates how an SSL connection test works.



The function tests an SSL handshake that takes place when an HTTPS connection is being built up. For the SSL handshake (to be tested), it is always required that the receiver (server) authenticates itself against the tenant (client).

Specify the following settings to configure the test:

SSL Connection Test Options

Attribute	Description
Host	Enter the host name of the receiver. The host name must **not** contain any path or schema for example, `https://`). In particular, you must **not** enter a URL as the host name.
Port	Enter the port that is to be used for outbound communication. Standard port is `443`.
Client Certificate Authentication (optional)	Choose this option if the client is to be authenticated against the receiver (server) during the SSL handshake (a mutual authentication). If you select this option, you can also specify the key to be used for the test (see below).

Attribute	Description
Private Key Alias (only if *Client Certificate Authentication* has been selected)	Enter the alias that identifies the relevant key pair for client certificate authentication. In case the receiver requires that during the SSL handshake also the tenant (client) is authenticated against the receiver (server), you can configure the connection test accordingly (selecting the *Client Certificate Authentication* option). In that case, provide the *Private Key Alias* (to indicate which key is to be used from the tenant keystore). In order for the tenant to be able to authenticate itself as the client against the receiver, a suitable key has to be available in the deployed keystore. With the *Private Key Alias* field you can narrow down further the check that a specific key is being used (for example, a key that matches the configuration settings in the corresponding adapter).
Validate Server Certificate	Allows you to validate the server certificate. During connection setup, the server is authenticated against the client using its server certificate. When you have selected the *Validate Server Certificate* option, the following checks are executed: • If the server certificate belongs to the server the client connects to • If the certificate is signed by an instance the client trusts If the *Validate Server Certificate*is not chosen (which is the default setting), both checks are not performed.

If the connection test was successful, the following information is displayed in a popup window:

- Host: Port
- Used alias (if a certificate-based connection has been tested)

When you have executed the test (and selected *Validate Server Certificate*), a subsequent screen provides you with the following options:

- *Download Certificates*
  Downloads the server certificate (the complete certificate chain) as compressed file to your local disk.
- *Details*
  Displays the details of the server certificate chain such as subject distinguished name (dn), issuer distinguished name (subjectDn), subjectsAlternativeNames, validUntil, and serialNumber.

> **Tip**
>
> That way you can compare the server certificate against the client certificate (from the tenant keystore). Doing this, you can check if the correct server root certificate has been imported into the tenant keystore.

Related Information

Client Certificate Authentication (Outbound) [page 222]

5.14.2 SSH Outbound Connection Test

When you have selected the SSH connection type, the test tool checks if the SSH outbound connection (configured in the related adapter) works correctly by executing a command (ls) that reads the directory structure on the associated SFTP server.

Depending on the chosen authentication, the following is checked by the test:

- If the server (host) is reachable for the tenant
- If the SSH server has accepted the user and the key (`id_rsa` or `id_dsa`) from the corresponding keystore deployed on the tenant
- If the configured `known_hosts` file deployed on the tenant contains the certificate of the SSH server

Specify the following settings to configure the test:

SSH Connection Test Options

Attribute	Description
Host	Enter the host name of the receiver. The host name must **not** contain any path or schema (for example, **`https://`**). In particular, you must **not** enter a URL as the host name.
Port	Enter the port that is to be used for outbound communication. Standard port is **`22`**.
Timeout (ms)	Specifies a timeout (in milliseconds) after which the connection to the server (host) should be terminated. The default value is 10.000 ms.

Attribute	Description
Authentication	There are the following options: • *Public Key* SFTP server authenticates the calling component based on a public key. • *User Credentials* SFTP server authenticates the calling component (tenant) based on the user name and password. To make this configuration setting work, you need to define the user name and password in a User Credential artifact and deploy the artifact on the tenant. • *Anonymous* SFTP server does not require any authentication of the calling component.
User Name (only if *Public Key* is selected as *Authentication*)	Enter the ID of the user under which the tenant calls the SFTP server.
Credential Name (only if *User Name/Password* is selected as *Authentication*)	Name of User Credential artifact deployed on the tenant. This artifact contains user name and password that are used for authentication at the server.
Check Host Key (only if *Public Key* or *User Credentials* is selected as *Authentication*)	Verifies the host key. **Note** It is recommended to use this option **with particular caution** when you have selected *User Credendials* as *Authentication* because in that case you take the risk that you forward the password to an unverified server.
Check Directory Access (only if *Public Key* or *User Credentials* is selected as *Authentication*)	Select this option if you like to check the access to the target directory. If you leave the *Directory* field empty, the default directory is your home directory. **Restriction** You can only specify sub-directories of the home directory.

When you have specified the settings for the test, select *Run* to execute the test.

If the connection test was successful, the following information is displayed in a popup window:

- Host: Port
- User Name
- The fingerprint of the host key

You have the following options

- *Copy Host Key*
  Copies the host key to your clipboard.

  > **Tip**
  >
  > When you use the connection test tool to get the host key by this way, it is strongly recommended that you always check if the fingerprint (displayed as result of the test) is identical to the one which has been provided to you by the SFTP server administrator through an independent channel.
  >
  > When you use the *Copy Host Key* option, the copied key is already in the format that is required for the associated known hosts file (deployed on the tenant). You can directly paste the host key into the known hosts file.

- *Details* (only when you have selected *Check Directory Access*)
  Displays details of the target directory configured on the SFTP server (which is the result of the ls command executed by the test tool on the server)..

Related Information

How SFTP Works [page 226]

5.14.3 SMTP Outbound Connection Test

You can perform an SMTP connection test, when a receiver Mail adapter has been configured.

SMTP Connection Test

Attribute	Description
Host	Host name of the receiver The host name must **not** contain any path or schema (for example, `https://`). In particular, you must **not** enter a URL as the host name.
Port	Port that is to be used for outbound communication (optionally contained in the *Address* field of the Mail adapter) Standard port is `587` for SMTP+STARTTLS or `465` for SMTPS (depending on the network protocol).
Encryption	Specify how encryption is to be used. Choose the same setting as configured in the corresponding Mail adapter.

Attribute	Description
Authentication	Choose which mechanism is to be used to authenticate against the server with a user name and password combination. Choose the same setting as configured in the corresponding Mail adapter.
Credential Name (only if *CRAM-MD5* or *User Credentials* have been chosen for *Authentication* in the corresponding Mail adapter)	Choose the name of a deployed credential to use for authentication. Choose the same setting as configured in the corresponding Mail adapter.
Validate Server Certificate	During connection setup, the server is authenticated against the client using its server certificate. When you have selected this option, the following checks are executed: • If the server certificate belongs to the server the client connects to • If the certificate is signed by an instance the client trusts If the *Validate Server Certificate*is not chosen (which is the default setting), both checks are not performed.
Check Mail Addresses	Select if sender and receiver addresses are to be checked during test. Enter sender and receiver email address (in *From* and *To* field).

If the connection test was successful, the following information is displayed:

- Host: Port
- Server version
- Selected Encryption mode
- Selected Authentication mode
- Credential name (if the corresponding options have been selected in the Mail adapter)
- The certificates
  You can download the certificates to your computer.

6 Web-Based Monitoring

An integration developer can use a Web user interface to check the status of messages and integration content artifacts for a tenant cluster.

To launch the Web-based monitoring application, access the URL provided by SAP in a Web browser.

The start page is subdivided into the following containers, each covering a specific task area. Each container contains a number of tiles.

Containers on Monitoring Start Page

Container	Allows you to ...
Monitor Message Processing	Monitor message processing on the tenant. Tiles in this section show the number and status of processed messages within a specified time window.
Manage Integration Content	Manage integration content for the tenant. Tiles in this section show the number and status of integration content artifacts (such as integration flows).
Manage Security	Manage security artifacts for the tenant. Tiles in this section show the number and status of security-related artifacts (such as keystores). The *Certificate-to-User Mappings* tile does not show any number. This tile is is only supposed to navigate to the related details page.
Manage Stores	Display the content of associated message queues.
Access Logs	Tiles in this section show information on system changes.
Manage Locks	This section allows you to display and manage lock entries that are created (in the in-progress repository) to avoid the same message being processed several times in parallel .
Configure B2B Integration	Provide an overview of number ranges related artifacts.

Each section contains tiles, which show filter settings and the number of messages/artifacts that correspond to the filter settings.

Clicking a tile opens a page with more information.

The start page is automatically refreshed every 5 seconds. The autorefresh ends after 5 minutes (indicated by a message).

> **Note**
>
> All times displayed on the Monitoring pages are local times (with regard to the client of the Web application).

Managing Tiles

You can customize the start page to fit your personal needs by adding new tiles (by clicking an empty tile containing a + symbol) or deleting existing ones. You can also rearrange the start page by dragging and dropping tiles to another location. For example, you can customize the start page so that it shows only data related to integration flows that are edited by you.

> **Note**
>
> You can rearrange tiles only within the same container.
>
> You cannot add, delete, or edit tiles in the following container types:
>
> - *Manage Security Material*
> - *Configure B2B Integration*
> - *Manage Storages*

> **Note**
>
> The personal settings are persisted in the client of the user. This means that you can log out, close the browser, and log on again to Web monitoring, and the previously specified user settings are kept.

To edit a tile, right-click the tile and choose *Edit*.

When you add or edit a tile, you can specify the following filter categories:

- For *Monitor Message Processing* tiles you can specify *Status*, *Time*, and the related *Integration Flow* for the messages to be displayed.
  In the dropdown list for the *Integration Flow*, the display name of the integration flow is shown. The tooltip for an entry shows the technical name and the version of the corresponding integration flow.
- For *Manage Integration Content* tiles, you can specify the *Status* and the *Type* of the integration content artifact to be displayed (the latter allows you to specify whether to display integration flows, OData services, value mappings, or all content types).

Related Information

Monitoring Message Processing [page 88]
Managing Integration Content [page 96]
Managing Security Material [page 100]
Managing Certificate-to-User Mappings [page 131]
Configure B2B Integration [page 143]
Monitoring Message Queues [page 138]

6.1 Monitoring Message Processing

The message monitor provides an overview of the messages processed on a tenant and allows you to display the details for individual messages.

You open the message monitor by clicking a tile in the *Monitor Message Processing* area.

Messages are displayed according to the filter settings of the tile.

Filter Settings

You can control which messages are displayed by changing the filter.

You can filter messages by *Time*, *Status*, *Artifact*, or by *ID*. The filter attributes have the following meaning:

Filter Attributes

Attribute	Description
Time	Allows you to select from the following predefined time intervals: • *All* • *Past Minute* • *Past Hour* • *Past 24 Hours* • *Past Week* • *Past Month* • *Custom* You can select the start and end time of the interval. The actually specified time interval is displayed on top of the message list. When you browse different pages of the message monitor, the time interval is not changed. You can modify the time interval only by changing the filter settings or by refreshing the message monitor (valid for all time intervals except of *Custom*).
Status	Allows you to filter messages according to their status. In the dropdown list you can select one of the following values as the status: • *All* • *Error* • *Failed* • *Retry* • *Completed* • *Processing* • *Escalated* For more information about the statuses, see the Related Links section below.

Attribute	Description
Artifact	Allows you to display messages associated with a specific artifact. The dropdown list contains all artifacts deployed on the tenant. You can filter for artifacts with a specific sequence of characters in their name or ID. The search is case-insensitive. The tooltip shows the technical name and the version of the artifact.
ID	Allows you to display messages associated with a specific MessageGuid or application ID. • MessageGuid Identifies the message uniquely. • Application ID Is set when an SAP_ApplicationID header element is specified in the associated integration flow in the Content Modifier step. When you filter by ID, the system checks if there is a message with the specified MessageGuid. If no such message is found, the system checks for messages with an Application ID that matches the entered value. Messages where MessageGuid and the application ID are identical (by coincidence) cannot be found by this filter attribute. Filtering messages by this attribute is helpful for support use cases (to do root cause analyses, for example).

Messages Table

The messages for the selected filter settings are displayed in a table (under *Messages*). If the number of filtered messages exceeds 50, the list is split into several pages (each page containing 50 messages at maximum). You can browse through the different pages selecting the corresponding navigation options (*Show first data page*, *Show last data page*, *Show previous data page*, and *Show next data page*).

Alternatively, you can enter a page number to navigate to a specific page of the list.

The following attributes are displayed for each message:

Message Attributes in Message Overview

Attribute	Description
Artifact Name	Display name of artifact (for example, name of integration flow that specifies the message processing). The tooltip shows the technical name and the version of the artifact.
Status	Status of end-to-end message processing.
Last Updated at	Time when message processing log has been updated at latest.

Attribute	Description
Log Level	The log level set for the message

For a selected message, the details are displayed to the right of the message table. The header area provides the following information about the selected message: *Processing Time*, *Receiver*, *Application Message ID*, and *Application Message Type*.

Below the header, the following sections contain detailed information about the selected message:

- *Status Details*
  Contains status information on the message. In case the message is **not** in status COMPLETED, the last error message is displayed.
- *Logs*
  Allows you to open the message processing log (MPL).
  The MPL displays structured information on the processing of a message, as well as the defined log level.
  In case the message contains attachments, you can open them in separate tabs.
  Any type of attachment that can be shown as text is supported. In particular,the following types of content can be displayed:
  - Plain text
  - XML
    XML attachments are displayed as formatted plain text (including tags).
  - Text files with comma-separated values
  - Text files with tab-separated values
    In such text files each entry is represented as one line of the text file.
  - HTML
    HTML attachments are displayed as plain text (including tags).
- *Artifact Details*
  Contains artifact information such as the artifact type and the artifact ID.
  If you choose *View Integration Flow*, the selected integration flow is opened in the design view.

Related Information

Message Status [page 36]
Message Processing Log [page 91]

6.1.1 Message Status

Message Status

Status	Description
COMPLETED	Message has been delivered to receiver successfully.
PROCESSING	Message is currently being processed.

Status	Description
RETRY	After during message processing an error occurred, a retry has been started automatically.
ERROR	During message processing an error occurred, but no automatic retry has been triggered. However, a manual retry can change the status.
ESCALATED	During message processing an error occurred and no retry has been triggered. For synchronous messages, an error messages is sent to the sender.
FAILED	Message processing failed, message has not been delivered to receiver, and no retries are possible. In other words: FAILED is a final status, message processing ultimatively has failed.

An aggregated message processing log (MPL) can have the following status values:

Status of Aggregated Messages

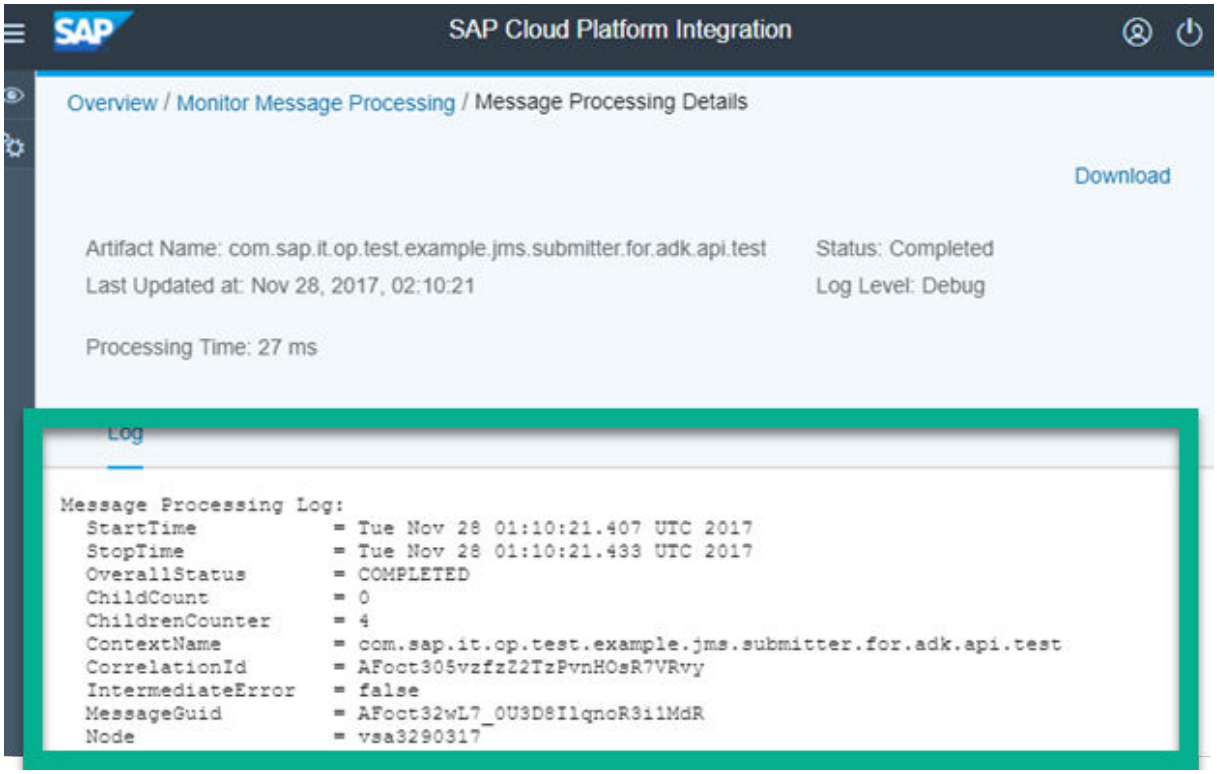
Status	Description
PROCESSING	Is set as soon as the aggregation process is started and remains as long as the aggregate is still open for further messages. Note that this status can in many situations be shown for quite some time (even days) – depending on the applied aggregation use case.
FAILED	Is set when the aggregated message (after aggregation process has successfully been finished) fails.
RETRY	Is set when the aggregated message (after aggregation process has successfully been finished) runs into an error and sending it is retried.

6.1.2 Message Processing Log

The message processing log displays structured information on the processing of a message.

The message processing log comprises a log header and log steps. In the log header you find more general information . The log steps show specific processing events and steps as well as further information, depending on the messages logged. This could be an `EscalationEvent` for example, which will trigger an alert showing up in the message processing log.

The log header shows the following properties:

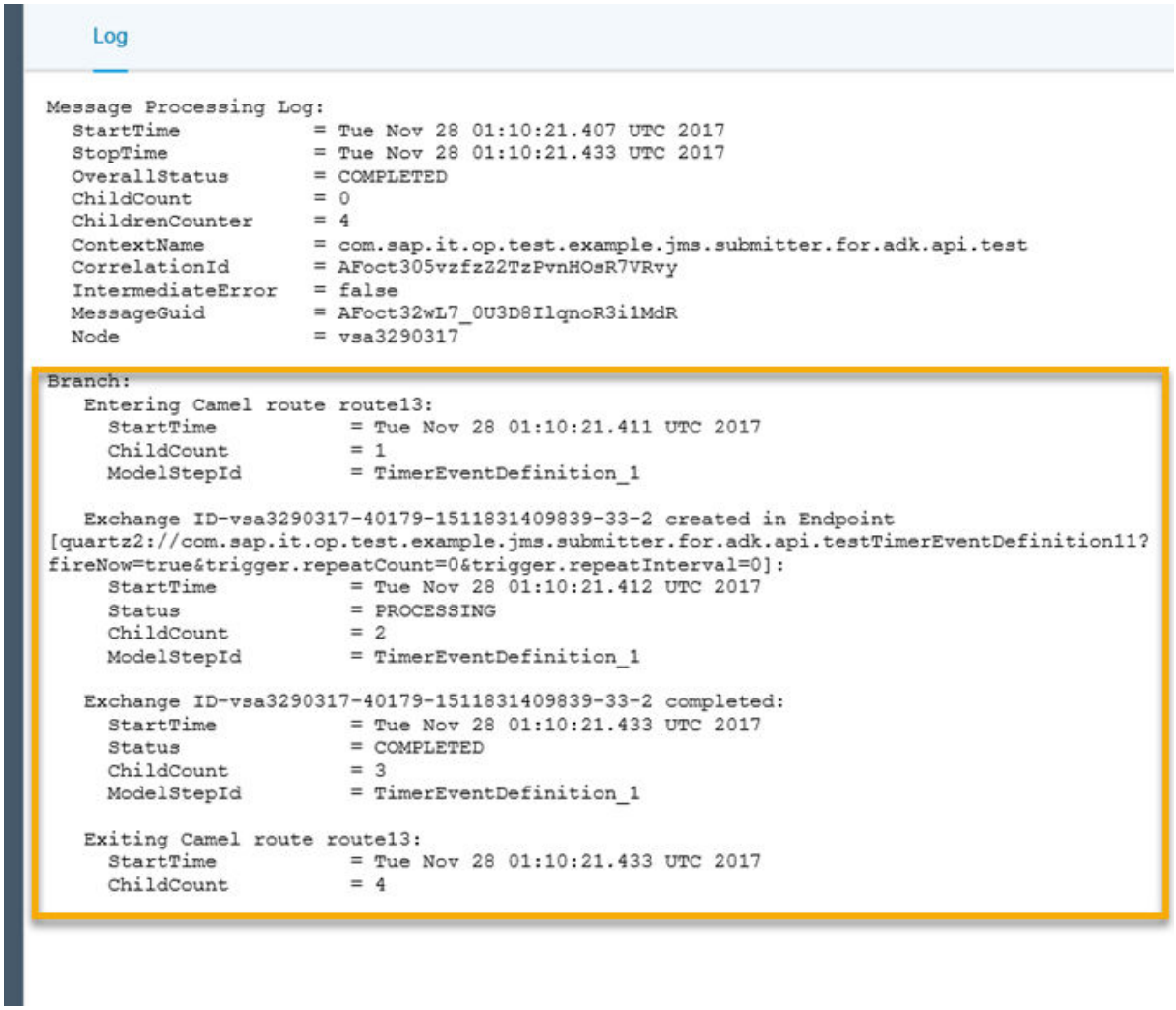


MPL Header

Property	Description
StartTime	Start of message processing.
StopTime	End of message processing.
OverallStatus	Specifies the end-to-end status of message processing and corresponds to the *Status* attribute in the Message Monitoring editor. For more information on statuses see **Message Status** in the **Related Information** section. .
ChildCount	Indicates the serial number of the current processing step.
ChildrenCounter	Specifies the total number of message processing steps.
ContextName	Integration flow name.

Property	Description
CorrelationId	ID that identifies correlated messages. Messages can be correlated, for example, when different integration flows on the same tenant communicate with each other. A correlation ID is a base64-encoded ID that is generated in this case by the first integration flow and stored in the message header. As part of the message header, the CorrelationId is then propagated across all related integration flows (that are in charge of processing the correlated messages).
CustomHeaderProperties	Can only be added by defining a script in the script API and the properties are displayed in the message processing log header. For more information see **Define Scripts** in the **Related Information** section.
Id	Is displayed in the MPL header if the ID has been previously defined as header property. For more information see **Headers and Exchange Properties** in the **Related Information** section.
IntermediateError	True if an error occurred (even temporarily) during message processing, or message processing took more than 1 minute. The header then contains an additional property: LastError showing the error type.
MessageGuid	Key that identifies the message uniquely in the database.
MessageType	Is displayed in the MPL header if the message type has been previously defined as header property. For more information see **Headers and Exchange Properties** in the **Related Information** section.
Node	Host name of the node that processed the message.
ReceiverId	Is displayed in the MPL header if the receiver ID has been previously defined as header property. For more information see **Headers and Exchange Properties** in the **Related Information** section.
SenderId	Is displayed in the MPL header if the sender ID has been previously defined as header property. For more information see **Headers and Exchange Properties** in the **Related Information** section.

The log steps show the following properties:



MPL Steps

Property	Description
Branch	Indicates that the following steps have been processed in the same branch. If a **Split** or **Multicast Step** is used, the steps belonging to one sub-message are grouped together within one **Branch**.
ModelStepId	This ID is used to specify relation between modeled step (in integration flow) and MPL entry. The integration flow model steps are fragmented in the Camel runtime environment into several processing steps. For more information see **Error Classification** in the **Related Information** section.

Property	Description
StepId	ID of the related integration flow step. It is assigned by the Camel framework.

> **Note**
>
> - The content of the message processing log varies according to the log level you have set.
> - The step properties displayed may vary from step to step, according to the message processing.

> **Note**
>
> - A length restriction to 1000 characters is defined for all string values and the system adds the following notification: "Property value was trimmed off. Max. value size is: 1000".
> - The MPL property names are limited to 100 characters. Any property name longer than 100 characters is cut off after 100 characters and the system adds the following notification: " Property name was trimmed off. Max. key name size is: 100
> - In case of `setBody` or `transform`, the length restriction applies to the payload which was written into the MPL in this particular case.The system adds the following notification:"Payload was removed due to security constraints. Use log attachments instead."

You can create attachments to the message processing log by using the **`Script API`**.

> **Note**
>
> The amount of MPL attachments which can be written is limited to 1 GB per 24 hours. If the limit is reached, MPL attachments will no longer be stored until the amount of MPL attachments written in the last 24 hours is again below 1 GB.

Related Information

Setting Log Levels [page 95]
Monitoring Message Processing [page 88]
Message Status [page 90]

6.1.2.1 Setting Log Levels

The log level for the message processing log specifies the granularity of information collected by the message processing log

You can set the log level for an integration flow scenario in the *Manage Integration Content* page of the *Web-based Monitor*.

Select your integration flow in the *Manage Integration Content* and set the log level for this integration flow in the *Log Configuration* section.

You can choose the following log levels:

- **`None`**: No Data is recorded during message processing and no data is shown in data monitoring.
- **`Info`**: Basic information is recorded during message processing. The header is always displayed and in case of failed messages, additional detailed information about the last 50 steps is retained and displayed in the message processing log..
- **`Debug`**: Detailed information is recorded for all steps during message processing. The header and additional information about the last 100 steps are displayed in the message processing log..
- **`Trace`**: Detailed information is recorded for all steps and in addition, the message content is tracked . The trace function expires after a certain time (default value: 10 minutes). After expiry the log level switches back to the log level set before. The recorded message content is also retained for a certain time (default value: 1 hour).

  > **Note**
  >
  > As the log level **`Trace`** is a high resource consuming feature, SAP can disable this function for a specific tenant, or adjust expiry and retention time.

> **Note**
>
> If you update an integration flow, that is, deploy an integration flow without deleting the predecessor version, the log level configuration is kept stable. If you undeploy an integration flow and then deploy a newer version, it is treated as a completely new integration flow which is logged with log level **`Info`**. Changes to the log level do not affect messages already in process and only apply to newly created messages after the time of modification.

> **Remember**
>
> We recommend to set log level **`Debug`** and **`Trace`** only in a development/test environment!

Related Information

Message Processing Log [page 91]
Monitoring Message Processing [page 88]

6.2 Managing Integration Content

The Manage Integration Content section provides an overview of integration content artifacts, such as integration flows, that have been deployed on the tenant.

In the *Manage Integration Content* area, a set of tiles is displayed that show the number of deployed integration content artifacts for a particular type (for example, Integration Flow) and with a particular status (for example,

Started). You can configure each tile in order to filter by a different artifact type or status, or both. To do this, position the cursor on a tile and choose *Edit* from the context menu.

To open the integration content monitor for a particular set of artifacts (as defined by the tile filter), click a tile.

When you click a tile, a list of integration artifacts is displayed (for the filter settings as defined for the tile). On the left side of the screen, the list of artifacts is shown in a table with the following attributes for each artifact:

- *Name*
  Below the name, the artifact type (for example, Value Mapping) is shown.
- *Status*

To customize the list of displayed artifacts, you can either search for specific artifacts by artifact name or ID **or** you can filter the table content by attributes such like Status or Type.

To sort and filter the content of the table, choose *Table Settings* (). On the subsequent screen, you can define how the table entries are to be sorted (by specifying an attribute and whether the entries are to be sorted for that attribute in ascending or descending order). You can also filter the table entries for certain attributes.

On the right side of the screen, more details about the artifact selected on the left side are shown. Depending on the artifact type, other functions can be accessed as well.

Integration Content Details

The header area provides the following information about the selected artifact:

- *Deployed on*
  Time when the artifact was deployed.
- *Deployed by*
  User who deployed the artifact.
- *Version*
  Version of the artifact, for example, the integration flow version.
- *ID*
- *Mode* (if applicable)
  Indicates whether it is configure-only content.

The header area also provides certain functions (the available functions depend on the type of the selected artifact):

- *Restart*
  Allows you to restart an artifact (for example, an integration flow that has been stopped on the node).
- *Undeploy*
- *Download*
  Allows you to download the artifact to your computer.
  You **cannot** download any content that has been classified by the content publisher as **configure-only**.

Below the header, the following sections contain detailed information about the selected artifact:

- *Endpoints*

Shows the URLs of the services exposed by the artifact for a sending application. You can copy the URL by selecting the Copy button  .
For example, if the artifact is an integration flow with a SOAP adapter configured as the sender channel, the endpoint URL specifies how the service exposed by the integration flow can be reached by a SOAP client.

> **Note**
>
> If the artifact is an OData service or an integration flow with an OData adapter configured as the sender channel, the endpoint URL is currently not available. You can construct the endpoint URL in the following format:
>
> ```
> <IFLMAP URL>/gw/odata/<OData Service Namespace>/<OData Service Name>;v=1
> ```
>
> The following table lists the main segments of the endpoint URL.

Endpoint URL Segment	Description
IFLMAP URL	Use the following procedure to select the IFLMAP URL: 1. If you have set up Integration Operations tooling in Eclipse, start Eclipse and proceed to step 3. 2. Install and configure Eclipse. For more information, see Installing and Configuring the Tool [page 10]. 3. In the *Node Explorer* view, choose *Show Properties* from the context menu of the IFLMAP node of your tenant. The *Properties* view appears with the *Node* tab selected by default. 4. In the *General* area, copy the value in the *URL* field. This is the IFLMAP URL.
OData Service Namespace	The namespace represents the logical grouping of the OData service. It is available in the metadata information of the OData service artifact. For more information, see .
OData Service Name	Ensure that the name of the OData service is capitalized in the endpoint URL. You can get the name of the OData service artifact from its metadata information. For more information, see .

- *Status Details*
  Shows the status of the artifact with regard to its usage at runtime (for example, if it is starting).
  If an error occurs during the lifecycle of an artifact, detailed information about the error is displayed. You can copy this information to the ticket you open to get the error solved.

  > **Note**
  >
  > Errors can occur in various steps during the lifecycle of an artifact. If you think of the artifact as an integration flow, steps are executed on the tenant management node (the user is connected to when deployment of the integration flow is triggered through the Web UI) as well as on the runtime node where the integration flow is to be processed at runtime.

Let's look at an example of an error that is related to the runtime node: If deployment of an integration flow is triggered, which requires a runtime node with a specific node profile, but no runtime node with this profile has been started in the cluster, the artifact cannot be distributed from the tenant management node to the runtime node. In this case, an error is displayed together with the information that no suitable runtime node has been started for this use case.

- *Artifact Details*
  The type of the selected artifact determines which kind of information is displayed.
  - Artifact Details for integration flows:
    The link *Monitor Message Processing* allows you to navigate to the Monitor Message Processing application.
    The link *View Integration Flow* allows you to navigate to the integration flow model.
  - Artifact Details for value mappings
    The link *View Value Mapping* allows you to navigate to the value mapping design screen.
  - Artifact Details for OData services:
    Name, *Namespace*, and *Version* of the OData service are displayed.

Related Information

Deploying an Artifact [page 70]
Artifact Statuses [page 99]

6.2.1 Artifact Statuses

An integration content artifact can have different statuses for each node.

The following statuses are possible.

Statuses

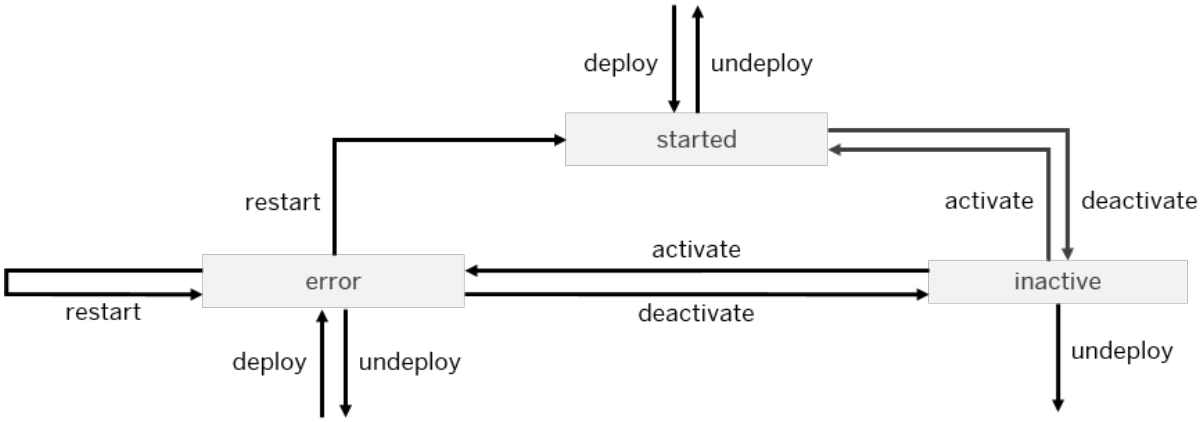
Status	Description
STARTED	The artifact is ready to be used on this node. An artifact with this status is in a final state.
INACTIVE	The artifact must not be used. An artifact with this status is in a final state.
ERROR	The artifact requires attention by the user (administrator). An artifact with this status is in a final state.
STARTING	The artifact is in the process of being deployed on the node(s) and, if deployed on the node(s), is being started. An artifact with this status is in an intermediate state.

Status	Description
STOPPING	The artifact is in the process of being stopped on the node(s) and, if stopped, is being undeployed. An artifact with this status is in an intermediate state.

The status is determined for each node (for example, the status is STARTED on two runtime nodes and ERROR on one runtime node).

Status information is displayed in the format: `status` (`number of nodes`). Different statuses for the nodes are separated with a space.

The following figure shows how the statuses are related to each other and which actions change the statuses.



6.3 Managing Security Material

The Manage Security Material area provides an overview of security-related artifacts.

You open the *Manage Security Material* area with the following actions:

Click the tile *Security Material*.

Security artifacts with the corresponding status (of the tile) are displayed.

Security Material Overview

A list of security material is displayed in a table. For each artifact, the following attributes are displayed:

Attributes of Security-Related Artifacts

Attribute	Description
Name	Display name of the artifact
Type	Possible values: • *Credentials* For *User Credentials* or *Secure Parameter* artifacts (a tooltip indicates the kind of the artifact). • *Keystore* • *SSH Known Hosts* • *PGP Public Keyring* • *PGP Secret Keyring*
Status	Displays the state with regard to the artifact deployment. This status indicates whether an artifact has been deployed successfully on a tenant. Possible values: • *Stored* **Note** Refresh the list to see the current status. • *Deployed* • *Error*
Deployed By	User who deployed the artifact
Deployed On	Time when the artifact was deployed

To sort and filter the content of the table, choose *Table Settings* (). On the subsequent screen, you can define how the table entries are to be sorted (by specifying an attribute and whether the entries are to be sorted for that attribute in ascending or descending order). You can also filter the table entries for certain attributes.

Actions

To create/deploy a new artifact, choose *Add* and select the artifact type (possible for the following artifact types: *Credential*, and *Known Hosts (SSH)*).

To edit and redeploy an existing artifact, select the artifact in the table and choose *Edit* (only supported for Credentials.

To download an artifact, select the artifact in the table and choose *Download* (not available for Credentials).

You can also delete an artifact (supported for all artifact types except *Keystore*).

Related Information

Deploying / Editing a User Credentials Artifact [page 102]
Deploying an SSH Known Hosts Artifact [page 105]
Deploying a Secure Parameter Artifact [page 106]
Deploying an OAuth2 Credentials Artifact [page 107]
Deploying a PGP Secret Keyring [page 103]
Deploying a PGP Public Keyring [page 104]

6.3.1 Deploying / Editing a User Credentials Artifact

To set up a connection using basic authentication or username token authentication, you have to specify the required attributes (for example, user name and password).

Context

User Credentials security artifact) on the corresponding tenant.You can specify basic authentication either for a connected SuccessFactors system (through the SuccessFactors adapter) or for general SOAP connectivity. To enable basic authentication for a runtime node, you specify the user credentials and deploy the attributes (in the form of a

> **Note**
>
> You can also edit and redeploy an existing artifact. To do that, select the artifact in the table under *Manage Security Material* and choose *Edit*.

Procedure

1. Click the tile in the *Manage Security Material* section.
2. Click *Add*.
3. Specify the following properties.

Option	Description
Name	A name for the artifact
Description	A description of the artifact (optional)
User	The user that calls the receiver system

Option	Description
Password	Password against which the user has to be authenticated

4. If you like to use the artifact to connect a SuccessFactors system, click *SuccessFactors* and specify the *Company ID* (which indicates the client instance used to connect to the SuccessFactors system).
5. Choose *OK*.

Results

When you refresh the *Manage Security Material* page, the new artifact is displayed (with Type *Credentials*) in the artifact table.

Next Steps

> **Note**
>
> When you have a User Credentials artifact, you also have to re-deploy and restart all integration flows that use this artifact.

6.3.2 Deploying a PGP Secret Keyring

This artifact contains the public and private key pair for the usage of Open Pretty Good Privacy (PGP). The private key enables the tenant to decrypt or sign messages.

Prerequisites

You have created a PGP secret keyring and stored it on your computer.

Context

Procedure

1. Select the tile *Security Material*.
2. Go to *Add* and select the option *PGP Secret Keyring*.
3. Browse for the secret keyring file on your computer and enter the passphrase of the PGP secret keyring. .

   > **Note**
   >
   > The password was defined when the PGP secret keyring was created. If the entered password does not match the original one, you cannot deploy the artifact and get an error message.

4. Click the *Upload* button to deploy your artifact.

   > **Note**
   >
   > To update the list click the refresh button.

   You can also select an artifact from the list and click *Download*.

   To undeploy an artifact select it from the list and click *Undeploy*. The system displays a warning and requires you to confirm the task. The undeployed artifact is removed from the list.

6.3.3 Deploying a PGP Public Keyring

This artifact contains the public key that enables the tenant to encrypt or verify messages using the Pretty Good Privacy (PGP) standard.

Prerequisites

You have created a PGP keyring file and stored it on your computer.

Context

Procedure

1. Select the tile *Security Material*.
2. Go to *Add* and select the option *PGP Public Keyring*.
3. Browse for the public keyring file on your computer.

4. Click on *Upload* button to deploy your artifact.

   > **Note**
   >
   > To update the list click the refresh button.

   You can also select an artifact from the list and click *Download*.

   To undeploy an artifact select it from the list and click *Undeploy*. The system displays a warning and requires you to confirm the task. The undeployed artifact is removed from the list.

6.3.4 Deploying an SSH Known Hosts Artifact

This artifact type specifies the known hosts file used when configuring secure connectivity based on SSH File Transfer Protocol (SFTP).

Context

The known hosts file contains the public keys and addresses of the trusted SFTP servers.

Procedure

1. Click the tile in the *Manage Security Material* section.
2. Choose *Add* > *Known Hosts (SSH)*.
3. Browse to the known hosts file on your computer.
4. Choose *OK*.

Results

When you refresh the *Manage Security Material* page, the new artifact is displayed in the table.

6.3.5 Deploying a Secure Parameter Artifact

Use the secure parameter artifact to deploy confidential data, for example, for custom adapters.

Context

You can store secure data like passwords in a secure store and use an alias name to access this data in an integration flow. The secure parameter artifact contains this alias name and confidential data.

Procedure

1. Click the tile in the *Manage Security Material* section.
2. Choose *Add* > *Secure Parameter*.
3. Specify the following attributes.

Option	Description
Name	Provide a name for the artifact. The artifact name is used as an alias for the confidential data assigned by this parameter.
Description	Provide a more detailed description of the artifact.
Secure Parameter	Enter the confidential value of the attribute.
Repeat Secure Parameter	Repeat the confidential value of the attribute.

4. Choose *OK*.

Results

When you refresh the *Manage Security Material* page, the new artifact is displayed in the table.

6.3.6 Deploying an OAuth2 Credentials Artifact

Many web servers can use OAuth 2.0 for authorization purposes. If you want to connect to a system that uses OAuth 2.0 authentication, you need to deploy an OAuth2 Credentials artifact using the following procedure.

Context

You can edit and deploy an OAuth2 Credentials artifact.

> **Note**
>
> Note that every time you edit an OAuth2 Credentials artifact, have to re-enter the Client Secret.

More information on OAuth: https://tools.ietf.org/html/rfc6749

Procedure

1. Click the tile in the *Manage Security Material* section.
2. Choose *Add* > *OAuth2 Credentials*.
3. Specify the following attributes.

Attribute	Description
Name	Name of the artifact that you want to deploy on the tenant
Description	Description of the artifact name you are deploying on the tenant
Authentication URL	URL of the OAuth authorization server that issues the access token
Client ID	ID of the client that you are connecting to
Client Secret	Secret key of the client that you are connecting to OAuth uses a multiple step authentication pattern: Client credentials (*Client ID* and *Client Secret*, as specified in the artifact) are used by the client application to initially request an access token. The access token is then used to authorize the client (for as long as the token is valid) to access the server's resources (for example, the resources that are used in the associated integration flow). In many OAuth scenarios, the access token is issued (or generated) by an authorization server.
Scope	Access rights that you are requesting from the service provider

4. Choose *Deploy*.

6.3.7 Managing Keystore Entries

The Keystore Monitor allows a tenant administrator to manage the tenant keystore and its entries (X.509 certificates and key pairs).

Context

> **Note**
>
> This feature is only available for node assembly version 2.29.* and higher.

Connections between a Cloud Integration tenant and a remote system can be secured using client certificate authentication (with X.509 certificates). The key pairs and certificates required to implement this authentication option are stored in a keystore.

A keystore typically contains entries that belong to the tenant administrator (customer) and entries that are owned by SAP. Each entry is uniquely identified by an alias.

The operations you can perform on a keystore entry depend on whether it is owned by SAP or the tenant administrator. You can, for example, only delete or back up entries that are owned by the tenant administrator.

Keystore entries are identified by an alias.

There are the following entry types:

- *Key Pair* entry
  Consists of a private key and its X.509 certificate chain.
  All private keys of a keystore are encrypted with the same password. This password is also used as the keystore password (for checking the integrity of the keystore). The keystore is never stored in the same database as the encrypted/signed application data. The password is stored in a separate database.
  The certificate chain typically consists of the public key certificate and the intermediate certification authority (CA) certificate with which the signature of the public key certificate can be verified.
- *Certificate* entry
  In many cases this is an X.509 root certificate.

> **Caution**
>
> Note the following restrictions when managing keystore entries:
>
> The maximum size of a keystore is 1 MB.
>
> - The 1 MB limit corresponds to around 1000 X.509 certificates.
> - A key pair with a chain of three X.509 certificates consumes about 3 KB, so if the keystore only contains key pairs of this type, then you can store around 300 key pairs in the keystore.
>
> A locking mechanism prevents different users from changing or deleting entries of the keystore at the same time. If a user is changing the content of a keystore, the keystore is locked for other users during that time.
>
> Private keys cannot be downloaded from a keystore.

Procedure

1. Choose the *Keystore* tile in the *Manage Security* section.
2. The *Current* tab shows the keystore entries available for the tenant.

   For each keystore, the single entries are displayed (by alias).

   The Keystore Monitor shows all keystore entries in a table. For each entry, the following attributes are displayed:

Attribute	Description
Alias	Uniquely identifies the keystore entry.
Type	Indicates whether the entry is a *Certificate* (X.509 certificate) or a *Key Pair* (with public and private key and an X.509 certificate chain). ○ Key Pair Consists of a private key and its X.509 certificate chain. ○ Certificate In many cases this is an X.509 root certificate.
Owner	The owner of the entry ○ *Tenant Administrator* The entry is owned by the tenant administrator on the customer side. ○ *SAP* The entry is owned by SAP.
Valid Until	Indicates the expiry date. If keys and certificates have expired, the date is highlighted in red.
Last Modified At	Indicates the date and time the entry was last modified.

> **Note**
>
> SAP-owned entries are indicated by a lock icon. You cannot change, delete. or back them up.

Next Steps

The Keystore Monitor also gives you the following options:

- Showing details of a keystore entry
- Uploading a keystore
  You can upload or add individual entries to an existing keystore. In the latter case, you can overwrite existing entries or keep them.
  SAP-owned keystore entries are indicated by a lock icon. You cannot change or delete them.
- Downloading the public content of a keystore or a single keystore entry

- Deleting a keystore entry
- Backing up keystore entries owned by the tenant administrator
- Restoring backed-up keystore entries

Related Information

Displaying Properties of a Keystore Entry [page 110]
Adding Keystore Entries [page 113]
Changing the Alias of a Keystore Entry [page 114]
Downloading a Keystore [page 115]
Downloading Single Keystore Entries [page 115]
Deleting Keystore Entries [page 117]
Backing Up Keystore Entries [page 118]
Restoring Backed-Up Keystore Entries [page 119]
Certificate Management [page 240]
Client Certificate Authentication and Certificate-to-User Mapping (Inbound) [page 215]
Client Certificate Authentication (Inbound) [page 218]
Client Certificate Authentication (Outbound) [page 222]
Managing the Lifecycle of Keys [page 120]
https://blogs.sap.com/2017/06/19/cloud-integration-keystore-monitor-now-available-for-tenant-administrator/

6.3.7.1 Displaying Properties of a Keystore Entry

Display properties of a selected keystore entry.

Context

There are the following entry types:

- *Key Pair* entry
  Consists of a private key and its X.509 certificate chain.
  All private keys of a keystore are encrypted with the same password. This password is also used as the keystore password (for checking the integrity of the keystore). The keystore is never stored in the same database as the encrypted/signed application data. The password is stored in a separate database.
  The certificate chain typically consists of the public key certificate and the intermediate certification authority (CA) certificate with which the signature of the public key certificate can be verified.
- *Certificate* entry
  In many cases this is an X.509 root certificate.

Procedure

1. Choose the *Keystore* tile in the *Manage Security* section.
2. Click the alias of a keystore entry to show the details.
3. The following attributes are shown for the selected keystore entry.

   In case, the keystore entry is part of a certificate chain, the certificate chain is displayed as a hierarchy tree at left.

   The following details are displayed for the selected keystore entry. When you click a certain element in the certificate chain, the details are adapted to show the details of this element.

   - The *Keystore* attribute indicates if the displayed entry is part of the currently active keystore (*Current*) or of the backed-up keystore (*Backup*).
   - The *Alias* is used to refer to a specific public key from a keystore.
   - The *Owner* can either by *SAP* or *Tenant Administrator*.

   Administration Attributes

Attribute	Description
Created By	User who created the certificate
Created On	Time when certificate has been created
Last Modified By	User that modified the keystore entry at the latest
Last Modified On	Time when the keystore entry has been modified at the latest

> **Note**
>
> The user shown in *Last Modified By* is anonymized (the string *SAP* is shown) in case the keystore is owned by SAP.

   General Attributes

Attribute	Description
Type	- *Key Pair* Consists of a private key and its X.509 certificate chain. - *Certificate* In many cases this is an X.509 root certificate.
Version	Certificate version
Serial Number	Serial number of certificate as provided by the issuer
Subject DN	Distinguished name of the entity associated with the public key of the certificate

Attribute	Description
Issuer DN	Distinguished name of the issuer (the certification authority that issued and signed the certificate)
Key Type	Cryptographic standard the keystore entry is using (for example, RSA)
Key Size	Number of bits (also referred to as *Key Length*)
Signature Algorithm	Algorithm used to create the signature of the key entry (for example, `SHA-512`)
Valid From	Start of validity period
Valid Until	End of validity period

The section *Fingerprints* shows different sequences defined to identify the public key. A fingerprint is generated out of the public key applying a hash function on the public key. For each fingerprint, the hash algorithm is also displayed (for example, `SHA-512`).

Next Steps

The following functions are available:

- You can delete the keystore entry in case it is owned by the Tenant Administrator.
- You can download the keystore entry.
- You can change the alias of the keystore entry in case it is owned by the Tenant Administrator.

Related Information

Changing the Alias of a Keystore Entry [page 114]

6.3.7.2 Adding Keystore Entries

Upload a new keystore. You can either replace an existing keystore completely or update an existing one by adding new entries. In the latter case, you can overwrite existing entries or keep them.

Context

> **Note**
>
> This feature is only available for node assembly version 2.29.* and higher.

> **Note**
>
> You cannot upload keystore entries with aliases starting with `sap_` or `hcicertificate`. Theses aliases are reserved for SAP-owned keystore entries.

Procedure

1. Choose the *Keystore* tile in the *Manage Security* section.
2. On the *Current* tab, above the table, choose *Add*.
3. Choose *Browse* and select the keystore on your local disk.
4. Enter the keystore password.
5. In the *Action* field, select one of the following options:

Option	Description
Add	Adds the entries from the uploaded keystore so that they are merged with the ones in the existing keystore.
Replace	Replaces the whole keystore with the uploaded one. **Note** Exception: SAP-owned entries are preserved as they cannot be changed or deleted by the tenant administrator.

6. If you have selected *Add* (in the *Action* field), you can specify whether existing entries are to be overwritten by uploaded entries with the same alias.
7. Choose *Add*.

Next Steps

A dialog is displayed showing a summary of the added, preserved, and overwritten entries. If any entries could not be uploaded, information is provided to explain why (for example, you tried to upload a keystore entry with an alias starting with `sap`).

6.3.7.3 Changing the Alias of a Keystore Entry

Context

An alias is a reference to a single keystore entry. You can use an alias to refer to and select a specific public key from a keystore.

> **Note**
>
> You can only change the alias of a keystore entry owned by the Tenant Administrator.
>
> The maximal length of a keystore alias is 250 characters.
>
> You cannot use aliases that are reserved for SAP (for example, starting with `sap`).

Procedure

1. Choose the *Keystore* tile in the *Manage Security* section.
2. Click the alias of a keystore entry to show the details.
3. Choose *Edit*.
4. Enter the new alias and choose *Save*.

Next Steps

Alternatively, you can change the alias of a keystore entry in the folowing way: Choose the *Keystore* tile in the *Manage Security* section and for a keystore entry choose *Edit* (under *Actions*).

6.3.7.4 Downloading a Keystore

Download a keystore to your local disk.

Context

> **Note**
>
> This feature is only available for node assembly version 2.29.* and higher.

Procedure

1. Choose the *Keystore* tile in the *Manage Security* section.
2. On the *Current* tab, choose *Download* (above the table).

Next Steps

The public part of the keystore is downloaded (in a file with the name `PublicContentKeystore.jks`).

The file is not password protected. You can open it with a third-party keystore editor (for example, KeyStore Explorer).

For private key entries, the certificate chains are resolved into single entries. The alias of the public key certificate is the same as the alias of the original key-pair entry. The intermediate certificate gets an additional suffix `_1` in the alias name. The root certificate gets an additional suffix `_2`.

6.3.7.5 Downloading Single Keystore Entries

Download a single keystore entry to your local disk.

Context

> **Note**
>
> This feature is only available for node assembly version 2.29.* and higher.

You can download a key, a single certificate, or a complete certificate chain (depending on the keystore entry type).

> **Note**
>
> You can download only public keystore content (no private keys).

Procedure

1. Choose the *Keystore* tile in the *Manage Security* section.
2. On the *Current* tab, select a keystore entry.
3. Choose one of the following options for the selected keystore entry.

   The options available depend on the type of keystore entry you have selected.

Option	Description
Key Pair **entry**	You can download the public key (X.509 format). ○ *Download Certificate* A file with the name `<alias>.cer` is downloaded. ○ *Download Certificate Chain* A file with the name `<alias>.p7b` is downloaded. The file contains the whole certificate chain. ○ *Download Public OpenSSH Key* (Only available for key pairs with the alias `id_rsa` or `id_dsa`, which indicates that this is an SSH key pair to be used for communication through SFTP.) **Note** A keystore can also contain keys for SFTP connections (SSH keys). Downloads the public key in OpenSSH format. You can store the downloaded `<alias>.pub` file on the connected SFTP server to enable public key authentication on the server. **Note** To configure public key authentication on certain SFTP servers, public keys need to be available in OpenSSH format. In such cases, use this download option. **Tip** Want to know how to get an SSH2 public key for the id_rsa/id_dsa key-pair of the tenant keystore?

Option	Description

	You can use ssh-keygen for this purpose (to be installed on Windows via Cygwin, for example).
	
	In ssh-keygen, perform the following command:
	
	`$ ssh-keygen -e -f id_rsa.pub -m RFC4716 > id_rsa.pub_ssh2`
	
	Example result file:
	
	`---- BEGIN SSH2 PUBLIC KEY ----`
	`Comment: "1024-bit RSA, converted by d023101@WDFN33785618A from OpenSS"`
	`AAAAB3NzaC1yc2EAAAADAQABAAAAgQCRB`
	`+UOPmnPF9W4cqn6h1z3V5izPFAOTCXSF4cfnw`
	`ZivjhV8ZAcKq6QwV/SyOXkXWp/wObjKgTXtiHngdJ0kyOQ+66Eleq/`
	`yhO4NDJ0QM3Vzv15`
	`IhL+eLYBtynk1ddF5l6kDLgSz1evA9F988wWKiz/`
	`vpI8DVjbY8HJjlQbE8wOSQ==`
	`---- END SSH2 PUBLIC KEY ----`

Certificate **entry**	*Download* A file with the name `<alias>.cer` is downloaded. It contains the base64-encoded certificate.

Next Steps

You can import the downloaded files into other keystores and open them using third-party tools (for example, KeyStore Explorer).

6.3.7.6 Deleting Keystore Entries

Context

> **Note**
>
> This feature is only available for node assembly version 2.29.* and higher.

Procedure

1. Choose the *Keystore* tile in the *Manage Security* section.
2. On the *Current* tab, select a keystore entry.
3. Choose *Delete* (next to the entry).

> ⚠ Caution
>
> You cannot delete SAP-owned keystore entries.

6.3.7.7 Backing Up Keystore Entries

Back up keystore entries owned by the tenant administrator.

Context

Procedure

1. Choose the *Keystore* tile in the *Manage Security* section.
2. On the *Current* tab, choose *Back Up* (above the table).

Next Steps

You can view the backed-up entries on the *Backup* tab.

Related Information

Restoring Backed-Up Keystore Entries [page 119]
https://blogs.sap.com/2017/08/14/cloud-integration-backuprestore-using-keystore-monitor/

6.3.7.8 Restoring Backed-Up Keystore Entries

Restore backed-up entries of the keystore (owned by the tenant administrator).

Context

You can restore keystore entries that you have backed up. For example, you can restore entries if you delete them from the active tenant keystore by mistake.

> **Note**
>
> Restored keystore entries from the keystore backup replace all existing entries of the tenant administrator. Therefore, be careful using this feature as you can delete by mistake existing keystore entries.

Procedure

1. Choose the *Keystore* tile in the *Manage Security* section.
2. On the *Backup* tab, the backed-up keystore entries are displayed in a table.

   The following attributes are displayed for each entry:

Attribute	Description
Alias	Uniquely identifies the keystore entry.
Type	Indicates whether the entry is a *Certificate* (X.509 certificate) or a *Key Pair* (with public and private key and an X.509 certificate chain). ○ Key Pair Consists of a private key and its X.509 certificate chain. ○ Certificate In many cases this is an X.509 root certificate.
Owner	The owner of the entry ○ *Tenant Administrator* The entry is owned by the tenant administrator on the customer side. ○ *SAP* The entry is owned by SAP.
Valid Until	Indicates the expiry date. If keys and certificates have expired, the date is highlighted in red.

Attribute	Description
Last Modified At	Indicates the date and time the entry was last modified.

3. Choose *Restore* (above the table).

   The selected entries are restored in the deployed tenant keystore.

Related Information

Backing Up Keystore Entries [page 118]
https://blogs.sap.com/2017/08/14/cloud-integration-backuprestore-using-keystore-monitor/

6.3.7.9 Managing the Lifecycle of Keys

To enable secure communication between the tenant and connected remote systems, the system keystore that is deployed on the tenant must contain the up-to-date keys owned by the tenant administrator and SAP.

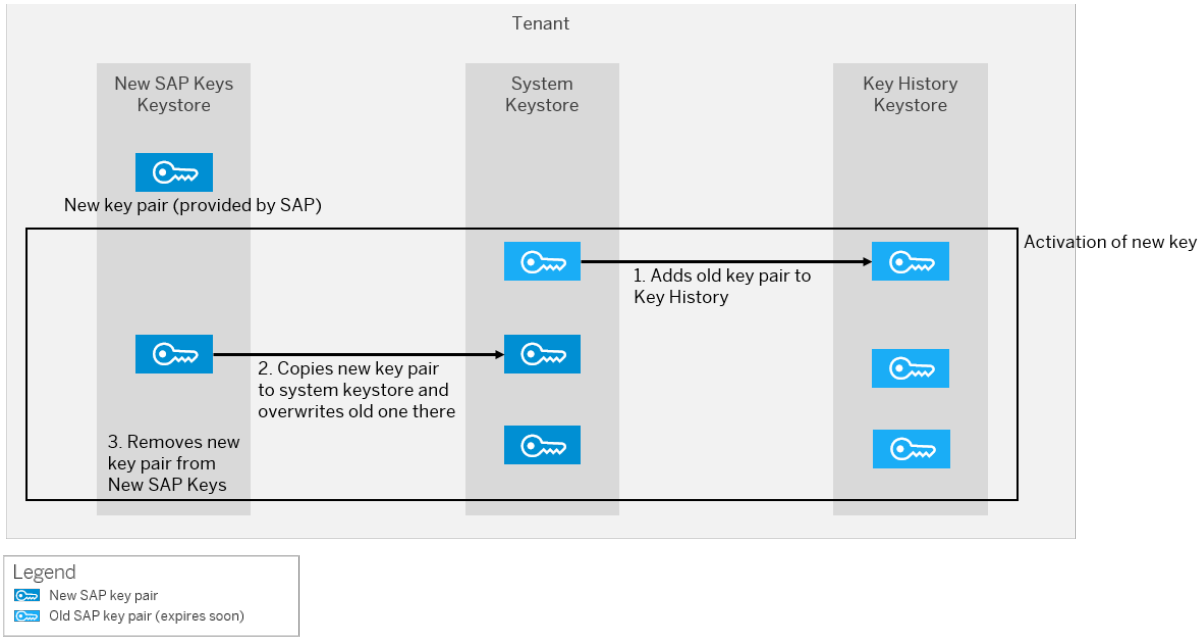
Context

SAP Cloud Platform Integration comes with a set of features that facilitate the tenant administrator's task of renewing keys provided by SAP on the tenant.

The following different keystore types enable the tenant administrator to renew keys in an efficient way:

- System keystore
  This keystore is located in the `system.jks` file provided with the tenant and contains all keys that are actively used by the deployed integration flows.
- New SAP Keys keystore
  This keystore contains keys already prepared by SAP before a key pair expires.
  As the key expiry date approaches, the tenant administrator can do the following:
  - Download the new SAP keys from the KeyRenewal keystore to share them with the administrators of the connected back ends
  - Activate the keys relevant for the tenant
- SAP Key History keystore
  This keystore contains old (expired) keys.
  If required, the tenant administrator can restore a key from the Key History keystore and use it to replace a key in the system keystore.

During the activation of an SAP key, the system performs three steps, as shown in the following figure.



1. The old key pair (which expires soon) is added by the system to the Key History keystore.
2. The old key pair (in the system.jks keystore) is overwritten by the new key pair from the New SAP Keys keystore.
3. In the New SAP Keys keystore, the new key pair is removed.

> **Note**
>
> To restore an old key pair, the tenant administrator can copy the key pair from the SAP Key History keystore to the New Keys keystore and activate it there.

Procedure

Related Information

Activating a New Key Pair Provided by SAP [page 122]
Restoring a Key Pair from the Key History [page 123]
Renewal of Keys Provided by SAP [page 201]

6.3.7.9.1 Activating a New Key Pair Provided by SAP

Activate a new key pair provided by SAP in order to replace an old key pair which is supposed to expire soon.

Context

> **Caution**
>
> Make sure that you activate a new SAP key pair as part of a dedicated key renewal process that you have agreed on with the administrators of all connected sender and receiver systems.
>
> For more information on the recommended processes, see the separate topics.

> **Note**
>
> Note regarding the case if you don't renew an SAP key prior to its expiration date:
>
> A system job makes sure that in this case the key is automatically activated (within a day after it has expired).
>
> This job checks daily if there are new entries in the New SAP Keys keystore and if an SAP key pair entry in the system keystore (system.jks) is expired. If this is the case, the job checks whether the New SAP Keys keystore contains an entry with the same alias. If this is the case, the key pair entry is activated (through the same sequence of steps as shown in the figure).

Procedure

1. Choose the *New SAP Keys* tile in the *Manage Security Material* section.
2. Select the entry and choose *Activate*.

Related Information

Renewal of Keys Provided by SAP [page 201]
Activating a New SAP Key Pair on the Tenant [page 202]

6.3.7.9.2 Restoring a Key Pair from the Key History

Restore a key pair from the SAP Key History.

Context

You use this process to restore an entry from the SAP Key History in the New SAP Keys keystore. In a subsequent step, you can then reactivate the key to make it available in the tenant's system keystore.

> ⚠ Caution
>
> Make sure that you activate a new SAP key pair according to the dedicated key renewal process that you have agreed with the administrators of all connected sender and receiver systems.
>
> For more information about the recommended processes, see the separate topics.

Procedure

1. Choose the *SAP Key History* tile in the *Manage Security Material* section.
2. Select the entry and choose *Add to New SAP Keys*.

   In the subsequent dialog, the old SAP alias name is proposed, but you have the option to overwrite it with a new SAP alias.

   > ➔ Tip
   >
   > You can specify a new SAP alias if you want to avoid overwriting already active key pairs with this step.

Related Information

Renewal of Keys Provided by SAP [page 201]
Activating a New SAP Key Pair on the Tenant [page 202]

6.4 Performing Connectivity Tests

You can test the connectivity to a receiver system.

Related Information

SSH Connectivity Tests [page 124]

6.4.1 SSH Connectivity Tests

Context

When you have selected the SSH connection type, the test tool checks if the SSH outbound connection reaches the associated SFTP server.

Depending on the chosen authentication, the following is checked by the test:

- If the server (host) is reachable for the tenant
- If the configured `known_hosts` file deployed on the tenant contains the certificate of the SSH server.

Procedure

1. In the *Monitor Message Processing* overview select *Test Connectivity*.
2. Select SSH.
3. Define the attributes for the test.

SSH Connection Test Options

Attribute	Description
Host	Enter the host name of the receiver. The host name must **not** contain any path or schema (for example, **`https://`**). In particular, you must **not** enter a URL as the host name.
Port	Enter the port that is to be used for outbound communication. Standard port is **`22`**.
Timeout (ms)	Specifies a timeout (in milliseconds) after which the connection to the server (host) should be terminated. The default value is 10.000 ms.

Attribute	Description
Authentication	There are the following options: ○ *Public Key* SFTP server authenticates the calling component based on a public key. ○ *User Credentials* SFTP server authenticates the calling component (tenant) based on the user name and password. To make this configuration setting work, you need to define the user name and password in a User Credential artifact and deploy the artifact on the tenant. ○ *Anonymous* SFTP server does not require any authentication of the calling component.
User Name (only if *Public Key* is selected as *Authentication*)	Enter the ID of the user under which the tenant calls the SFTP server.
Credential Name (only if *User Name/Password* is selected as *Authentication*)	Name of User Credential artifact deployed on the tenant. This artifact contains user name and password that are used for authentication at the server.
Check Host Key (only if *Public Key* or *User Credentials* is selected as *Authentication*)	Verifies the host key. **Note** It is recommended to use this option **with particular caution** when you have selected *User Credendials* as *Authentication* because in that case you take the risk that you forward the password to an unverified server.
Check Directory Access (only if *Public Key* or *User Credentials* is selected as *Authentication*)	Select this option if you like to check the access to the target directory. If you leave the *Directory* field empty, the default directory is your home directory. **Restriction** You can only specify sub-directories of the home directory.

6.4.2 SMTP Connectivity Tests

You can perform SMTP connectivity tests to check the settings required for configuring the receiver mail adapter.

SMTP Connection test

Attribute	Description
Host	Enter the host name of the receiver. The host name must **not** contain any path or schema (for example, `https://`). In particular, you must **not** enter a URL as the host name.
Port	Choose or enter the port that is to be used for outbound communication. Standard port is `587` or `465` depending on the network protocol .
Protection	Specify which protection is to be used.
Authentication	Choose which mechanism is to be used to authenticate against the server. Possible values are: • *None* No authentication is attempted. No credential can be chosen. • *Encrypted User/Password* The user name and password are not sent in plain text to the server. This authentication mechanism is secure even without a secure connection. • *Plain User/Password* The user name and password are sent in plain text. You should only use this option together with SSL or TLS, as otherwise an attacker could obtain the password.
Credential Name (only if *User Credentials* has been chosen for *Authentication*)	Choose the name of a deployed credential to use for authentication.
Check Mail Addresses	Enter sender and optionally receiver email address.

If the connectivity test was successful, you get the information about the different checks. Choose *Download* if you want to save the certificates on your computer.

6.4.3 TLS Connectivity Test

When you have chosen the TLS connection the test tool checks the following:

- If the receiver (host) is reachable for the tenant
- If the keystore is deployed correctly and contains those keys that are required for the specified authentication method during TLS handshake

To perform the TLS connectivity test you need to specify the following settings:

TLS Connectivity Test Options

Attribute	Description
Host	Enter the host name of the receiver. The host name must **not** contain any path or schema for example, `https://`). In particular, you must **not** enter a URL as the host name.
Port	Enter the port that is to be used for outbound communication. Standard port is `443`.
Authenticate with Client Certificate (optional)	Choose this option if the client is to be authenticated against the receiver (server) during the TLS handshake (a mutual authentication). If you select this option, you can also specify the key to be used for the test (see below).
Alias (only if *Authenticate with Client Certificate* has been selected)	Enter the alias that identifies the relevant key pair for client certificate authentication. In case the receiver requires that during the TLSL handshake also the tenant (client) is authenticated against the receiver (server), you can configure the connection test accordingly (selecting the *Authenticate with Client Certificate* option). In that case, provide the *Alias* (to indicate which key is to be used from the tenant keystore). In order for the tenant to be able to authenticate itself as the client against the receiver, a suitable key has to be available in the deployed keystore. With the *Alias* field you can narrow down further the check that a specific key is being used (for example, a key that matches the configuration settings in the corresponding adapter).

Attribute	Description
Validate Server Certificate	Allows you to validate the server certificate. During connection setup, the server is authenticated against the client using its server certificate. When you have selected the *Validate Server Certificate* option, the following checks are executed: • If the server certificate belongs to the server the client connects to • If the certificate is signed by an instance the client trusts If the *Validate Server Certificate* is not chosen (which is the default setting), both checks are not performed.

If the test is successful, you get a response that the host has been successfully reached and which certificates were used. Choose *Download* if you want to save these certificates on your computer.

6.4.4 IMAP Connectivity Tests

When you have chosen the IMAP (Internet Message Access Protocol) connection the test tool checks the following:

- If the receiver (host) is reachable for the tenant

IMAP Connectivity Test Options

Attribute	Description
Host	Enter the host name of the receiver. The host name must **not** contain any path or schema for example, `https://`). In particular, you must **not** enter a URL as the host name.
Port	Enter the port that is to be used for outbound communication. Standard port is `143` or `993`
Protection (optional)	Select which protection is be used.

Attribute	Description
Authentication	Choose which mechanism is to be used to authenticate against the server. Possible values are: • *Encrypted User/Password* The user name and password are not sent in plain text to the server. This authentication mechanism is secure even without a secure connection. • *Plain User/Password* The user name and password are sent in plain text. You should only use this option together with SSL or TLS, as otherwise an attacker could obtain the password.
Credential Name	Select credential name. When you have selected the *Validate Server Certificate* option, the following checks are executed: • If the server certificate belongs to the server the client connects to • If the certificate is signed by an instance the client trusts If the *Validate Server Certificate* is not chosen (which is the default setting), both checks are not performed. If the response was successful, you the certificates are displayed and you can dowload the information.
• *List Folders* and or for • *Check Mailbox Content*	If you check *List Folders*, you get a list of mail folders. If you check *Check Mailbox Content* the folders are checked and the total number of mails and the number of unread mails is displayed.

6.4.5 POP3 Connectivity Tests

When you have chosen the POP3 (Post-Office_Protocol) the test tool checks the following.

IMAP Connectivity Test Options

Attribute	Description
Host	Enter the host name of the receiver. The host name must **not** contain any path or schema for example, `https://`). In particular, you must **not** enter a URL as the host name.

Attribute	Description
Port	Enter the port that is to be used for outbound communication. Standard port is **110** or **995**
Protection (optional)	Select which protection is be used. **Note** STARTTLS is not supported for port 995. For port **110**, STARTTLS is supported and checked.
Authentication	Choose which mechanism is to be used to authenticate against the server. Possible values are: • *Encrypted User/Password* The user name and password are not sent in plain text to the server. This authentication mechanism is secure even without a secure connection. • *Plain User/Password* The user name and password are sent in plain text. You should only use this option together with SSL or TLS, as otherwise an attacker could obtain the password.
Credential Name	Select credential name. When you have selected the *Validate Server Certificate* option, the following checks are executed: • If the server certificate belongs to the server the client connects to • If the certificate is signed by an instance the client trusts If the *Validate Server Certificate* is not chosen (which is the default setting), both checks are not performed. If the response was successful, you the certificates are displayed and you can dowload the information.
Check Mailbox Content	If you check *Check Mailbox Content* the mailbox is checked and the total number of mails in the inbox is displayed.

6.5 Managing Certificate-to-User Mappings

The Manage Security Material area provides an overview of security-related artifacts. It also provides access to all certificate-to-user mappings defined for the tenant.

You can display all certificate-to-user mappings deployed on the tenant by clicking the *Certificate-to-User Mapping* tile under *Manage Security Material*.

Certificate-to-User Mapping Overview

A list of certificate-to-user mappings is displayed in a table. For each artifact, the following attributes are displayed:

Attributes of Certificate-to-User Mapping Artifacts

Attribute	Description
User Name	Name of the user to which the certificate is to be mapped
Subject DN	Subject distinguished name of the certificate (identifies the entity that is associated with the certificate)
Issuer DN	Issuer distinguished name of the certificate (identifies the issuer who has signed the certificate)
Serial Number	Unique serial number of the certificate
Valid Until	Time when the certificate expires
Last Modified by	Name of user who has modified the certificate-to-user mapping the last time
Last Modified on	Date and time when certificate-to-user mapping has been modified the last time

> **Note**
>
> Limit for certificate-to-user mapping: 1 MB (corresponds to about 1000 X.509 certificates)

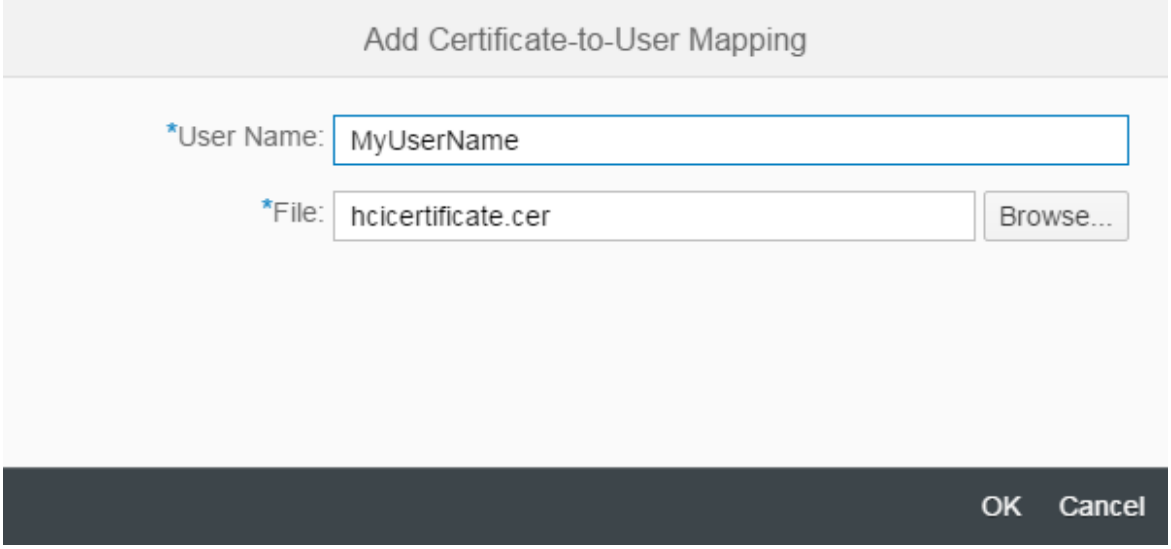
To sort and filter the content of the table, choose *Table Settings* (). On the subsequent screen, you can define how the table entries are to be sorted (by specifying an attribute and whether the entries are to be sorted for that attribute in ascending or descending order). You can also filter the table entries for certain attributes.

The search allows you to filter specific certificate-to-user mappings by providing parts of their name.

Actions

To add a new artifact, perform the following steps:

1. Choose *Add*.
2. Specify the *User Name* and (next to the *File* field) click *Browse* and search for the certificate file (.cer file) on your computer.



3. Choose *OK*.

To edit an existing Certificate-to-User Mapping artifact, perform the following steps:

1. Select the artifact in the table and choose *Edit*.
2. You have the options to change the User Name, specify (select) another certificate, or to change both attributes.

You can also delete an artifact.

Related Information

X.509 Certificates [page 240]
Client Certificate Authentication and Certificate-to-User Mapping (Inbound) [page 215]

6.6 Managing Data Stores

Context

The *Manage Stores* section provides an overview of storages on the tenant, which are temporarily used to persist data of different kind during message processing.

Procedure

1. To open the *Manage Data Stores* view, choose the *Data Stores* tile.
2. On the left pane of *Manage Data Stores*, the list of all the available data stores is shown with the following attributes for each store:

   > **Note**
   >
   > The search allows you to filter specific data stores by providing parts of their name or Integration Flow name.

Attribute	Description
Data Store Name	Name of the storage on the tenant.
Visibility	Displays the transient data store that can either be shared across all integration flows deployed on the tenant (global data store) or only be used by one integration flow (local data store).
Number of Entries	Displays the number of entries available in the data store.
Number of Overdue Entries	Displays the number of entries that are over

3.

Table Settings	Description
Sort By	Allows you to sort the messages either in ascending or descending order. You can select one of the following sorting criteria: ○ *Name* ○ *Visibility* ○ *Total Number of Entries* ○ *Number of Overdue Entries*
Filter By	Allows you to sort by message using the following sorting criteria: ○ *Status*: Sorts messages with or without overdue entries (you can select either one or both options). ○ *Visibility*: Sorts the data store by the global or technical name of the integration flow. When you sort by global visibility, all the entries are shown at the beginning or at the end of the list. All other entries are alphabetically sorted.

4. On the right side of the screen, details about the data store selected from the left side are shown.

> **Note**
>
> The *Delete* option (above the table of entries) allows you to delete all entries from the data store.

5. Use the following options available on the header above the table to manage the entries:

Function	Description
Search for ID	You can filter the list by providing the entry ID in the search field. The list displays entries that contains the search strings.
Delete	Deletes one or multiple data store entries.
Download	Downloads an entry to your computer. You can download only one entry at a time.
Reload Content	Refreshes the data store entry list.
Table Settings	Allows you to sort and filter the entries.
Multiselect Mode	You can use this option to delete multiple entries or to download a single entry .

6.7 Monitoring Audit Log

The audit log contains information on system changes. These events can be for example the deployment of an integration flow as well as a configuration change.

You view the audit log by clicking the tile *Audit Log* in the *Access Logs* area.

> **Note**
>
> To view the content of the log, you need the authorization **`AuditLog.Read`** .

You can control the display of the messages by adjusting the filter setting *Time Range*.

You can choose from the following preset time ranges:

- All
- Past Hour
- Past 24 Hours
- Past Week
- Custom

> **Note**
>
> The audit log data retention time in the database is 30 days.

You can retrieve the following information from the audit log list and filter the entries by *Object Name*, *User* or *Source*

Audit log List

Attributes	Description
Time	Displays the time of the event.
Action	Displays the action performed on the system such *Create* or *Delete*
Object Type	Displays the object type such as *IntegrationFlow* on which the action was performed on.
Object Name	Displays the object name such as the integration flow name.
User	Displays the user who triggered the action.
Source	Displays the IP address of the source that issued the action.

> **Note**
>
> If an SAP user triggers the changes, *User* and *Source* are displayed as *SAP* in the audit log list.

You can also sort the audit log list by *Time*, *Action*, *Object Type* or *Object Name*.

> **Note**
>
> The audit log retrieves a maximum of 1000 entries from the data base. If you have more than 1000 entries in the selected time range, you will be prompted to adjust your filter settings accordingly .

Related Information

Tasks and Required Roles [page 18]

6.8 Monitoring System Log Files

This section contains information on system log files. These log files can be either HTTP access files or Cloud Platform default trace files.

You view the system log files by clicking *System Log Files* in the *Access Logs* area.

You can retrieve the following information from the system log file list:

Log Files

Attributes	Description
Name	Displays the log file name
Log Type	Displays 2 different log types the CP default trace or the HTTP access log
Updated At	Displays the date of the last update
Size	Displays the file size
Actions	You can either download the file by selecting  or get the URL by selecting  .

You can filter the log files by names and sort the list either by *Name* , *Updated At*, or file *Size*. You can either download a specific file or get the file URL. The file URL allows you to send it per mail to allow further analysis on another computer for example. In both cases, the system provides a packed log file (.zip file).

> **Note**
>
> The log file retention time is 7 days.

If you select *Collections*, you get the most recent log files for each runtime node. You can either download the collection or get the URL and in both cases the system provides packed log files (.zip files).

6.9 Managing Variables

The *Variables* view allows you to monitor variables used in integration flows.

Choose the *Variables* tile in the *Manage Store* section. You get an overview of the existing variables, with the following attributes.

Table Settings	Description
Name	The variable name is defined in the integration flow.
Visibility	A variable can be globally visible across all deployed integration flows of the tenant or be used only by one integration flow.
Integration Flow	Displays the ID of the integration flow the variable is used in.
Updated At	Shows date and time when the variable content was last updated.
Retain Until	Displays date and time until the variable is still available. The retention time is updated along with any update of the variable.
Actions	You can download the variable content or delete the variable.

You can filter in the table either by variable name or integration flow.

> **Note**
>
> To view or download variables you need the authorization `ESBDataStore.readPayload`.

By clicking on the variable name in the table, you can see its content. If the variable content is not defined as a string value, its content cannot be displayed and a message is shown. You can also download the variable by choosing *Download*. If you choose to save the variable, the system creates a `.zip` file, containing the header properties file.

> **Note**
>
> If you want to delete a variable, check that this variable is no longer in use, as there is no " where used list" available.

Related Information

Tasks and Required Roles [page 18]

6.10 Monitoring Message Queues

Certain adapters allow you to store messages in queues. Using the Web UI, you can monitor queues that are active for a tenant.

To open the queue monitoring application, open the Web UI for your tenant and choose *Monitor*.

If one or more queues are active for your tenant, a *Message Queues* tile is displayed (under *Manage Storages*).

> **Note**
>
> Message queues are only supported by the following adapter types:
>
> - AS2 adapter
> - JMS adapter
>
> Click the tile to access more information on the message queues.

Click the JMS message ID to jump directly to the message processing log for the message, where you can find the integration flow name, the time the message was sent, and the details of message processing.

Message Queue Overview

Below the header, the used capacity is displayed.

It shows you how much of the available memory is already used, e.g 800.5/1000 kB.

If you reach 80% of the available memory, the capacity row turns yellow. If you reach 98% of the available memory, the row turns red.

All message queues that are active for the tenant are displayed in a table. For each queue, the following information is displayed:

Message Queue Overview

Attribute	Description
Queue Name	Name of JMS queue
Number of Messages	Number of messages stored in queue
Size (in MB)	Size of the queue (in MB)

To sort and filter the content of the table, choose *Table Settings* (). On the subsequent screen, you can define how the table entries are to be sorted (by specifying an attribute and whether the entries are to be sorted for that attribute in ascending or descending order). You can also filter the table entries for certain attributes.

The search allows you to filter specific queues by providing parts of their name.

You can select a queue and perform the following actions (under *Actions*):

- *Retry*
  Triggers retry of all messages in the selected queue.
- *Delete*
  Deletes the selected queue and all messages stored in the queue.

  > **Note**
  >
  > After having performed this action, you need to revisit and redeploy the integration flow. After having deleted a queue, the integration flow cannot write any more messages into the queue and, therefore, cannot process the messages correctly

- *Where-Used*
  You use this function to display the integration flows in which a queue is used. If you select the link to the integration flow, it opens in read-only mode and you can check the scenario that is using the queue. During the operation of your scenarios you may find that messages are piling up in one queue, and you want to get more details so that you can analyze why the messages are not being processed.
- *Check*
  You use this function to show unused and missing queues.
  The check results show the following:
  - Queues that are not used in any of the deployed integration flows
  - Queues that are referenced by integration flows but do not actually exist because they have been deleted by mistake

  You can delete any queues that are not needed, and generate any missing queues by redeploying the integration flow.

Message Queue Details

To display more details on a message queue, click on the related table row.

The message queue details view shows all messages of a queue in a table. For each message, the following information is displayed in a table:

Message Queue Details

Attribute	Description
JMS Message ID	
Message ID	Message ID that identifies the associated message processing log To access the associated message processing log, click on the Message ID.

Attribute	Description
Status	The following values are possible: • Waiting The message is still waiting to be processed • Overdue The message has not yet been processed and the deadline (Due At Date) has passed • Failed An attempt was made to process the message but an error occurred • Blocked The message was involved in multiple node crashes and was therefore not processed. The message is not automatically restarted as it was probably this message that caused the node to crash. You can see that the status is *Blocked*, and also that the *Retry At* column is empty. Please check the message size and the integration flow. The message may be too large or there could be an error in the integration flow. You can then either delete the message or perform a manual retry in the Queue Monitor. Make sure that the problem that caused the crash in the integration flow has been resolved. Maybe the size of the virtual machine (VM) has to be increased.
Due At	Due date for the message The due date is calculated as follows: The number of days configured as the *Retention Threshold for Alerting* in the adapter that the message originates from is added to the date on which the message is created. Once the due date is reached, the status of the message is set to *Overdue*.
Created At	Date when message has been stored in the queue
Retry Count	Number of retries of the message
Next Retry On	Date when next retry is scheduled
Retain Until	Date until which the message is retained. Once this date is reached, the message is automatically deleted.

You can perform the following actions for a selected message:

- *Retry*
  Triggers retry of the selected message.
- *Delete*
  Deletes the selected message.
- *Download*
  Downloads the message with attachments as a zip file.
  The zip file has four entries with the following names:
  - `attachments`
    Folder that contains message attachments (as `.txt` files)
  - `body`
    Contains the message body.
  - `properties.prop`

Contains the exchange properties.

- `headers.prop`
  Contains the headers.

This file was generated from a JMS message with an attachment called info.txt (`attachments`), a message body (`body`), exchange properties (`properties.prop`), and headers (`headers.prop`).
If a JMS message has several attachments, a zip entry is generated for each attachment. The names of the attachment entries have the following form:
attachment_<File Name | Content ID | UUID>
<File Name | Content ID | UUID> means that either the file name, the content ID header, or a UUID is used as the name component, depending on whether the value for the file name or the content ID is available.
If there is no attachment, the zip file does not contain an attachment entry.
The naming convention for the entry with the message body is body_<Content ID>, where _<Content ID> is only included if there is a content ID header for the body.
The entries exchange.properties and header.properties are only included if the JMS message has properties or headers.

Required Authorizations

To use the API to access the data store, you need to assign the following authorization groups to your user:

To use the API to access the data store or message queues, you need to assign the following authorization groups to your user:

- `AuthGroup.Administrator` or `AuthGroup.IntegrationDeveloper`
  To display the message overview
- `AuthGroup.Administrator`
  To delete messages
- `AuthGroup.Administrator` or `AuthGroup.IntegrationDeveloper`
  To trigger message retries
- `AuthGroup.BusinessExpert`
  To download a message

Related Information

https://blogs.sap.com/2017/10/04/cloud-integration-checks-in-jms-message-queue-monitor/
https://blogs.sap.com/2017/07/17/cloud-integration-configure-dead-letter-handling-in-jms-adapter/

6.11 Managing Locks

This section allows you to display and manage lock entries that are created (in the in-progress repository) to avoid the same message being processed several times in parallel (for example, by different runtime nodes).

> **Tip**
>
> Example:
>
> Several runtime nodes try to read a file from an SFTP server (through SFTP sender channels).
>
> To prevent double processing, a lock entry is written to the in-progress repository each time a file is processed by a runtime node. As long as this lock entry exists, no other component can access the file. After message processing, the lock is removed by the runtime.

In certain situations (for example, a runtime node crashes because of an out-of-memory error), the message is retried after the node is restarted until the expiration time is reached. In this case, lock entries could remain in the in-progress repository and block subsequent message processing. You can tuse the *Manage Locks* view to analyze the situation and manually delete lock entries, if required, to reprocess the message.

If you choose the *Manage Locks* tile in the *Monitor Message Processing* application, a list of locks is displayed. The following information is shown for each lock entry:

Attribute	Description
Component	Component that wrote the lock entry Currently, only SFTP adapters can write locks.
Source	Component that caused the lock Example for SFTP: `user_sftp@ld1234.mycompany.corp`
Entry	Content of the lock entry Example for SFTP: `directory1/dir2/test.xml` If the lock relates to an SFTP connection, the *Entry* field shows a file name (of the file that the SFTP adapter tries to read from the SFTP server). This parameter provides a link to the message in the Managing Message Queues monitor. To access the associated JMS Message, click *ID*. The Queue Monitor opens and loads the JMS Message. This is only possible for JMS locks.
Created at	Time when the lock was created This is the time when a runtime node tried to process the message for the first time.

Attribute	Description
Expires at	Time when lock entry expires If the message is retried but cannot be processed successfully before the expiration time, processing is stopped. In this case, the lock entry has to be removed manually to enable further processing.

You can also search for table entries (search field).

To sort and filter the content of the table, choose *Table Settings* (). On the subsequent screen, you can define how the table entries are to be sorted (by specifying an attribute and whether the entries are to be sorted for that attribute in ascending or descending order). You can also filter the table entries for certain attributes.

To remove the lock entry and retrigger message processing, select the entry and choose *Release*.

> **Caution**
>
> Before releasing a lock entry, make sure you do a careful problem analysis. In particular, make sure that you have understood how the lock entry in question relates to the actual problem you are trying to solve. Careless usage of the release lock function may lead to data inconsistencies or other serious problems.
>
> For example, in the case of an SFTP connection the lock entry is a file name (including the file path). In this case, check whether the problem is related to the file that is to be processed through SFTP.
>
> Another indicator is the duration of the lock (time that has passed since the time specified under *Created at*).

Related Information

https://blogs.sap.com/2017/07/17/cloud-integration-configure-dead-letter-handling-in-jms-adapter/

6.12 Configure B2B Integration

The configure B2B integration area provides an overview of number ranges related artifacts.

You open the *Configure B2B Integration* area with the following actions:

Click the tile *Number Ranges* in the *Configure B2B Integration* area.

Number range artifacts with the corresponding status (of the tile) are displayed.

Number Ranges Overview

While sending out a document in case of EDI processing, a unique interchange number must be added to each document. In order to add such an interchange number you can use the Number Range Object. In case of EDIFACT the outgoing EDIFACT messages should have Interchange Control Reference with length 1 to 14 digits, with minmum length being 1 digit and maximum length ranging to 14 digits.

A list of number ranges is displayed in a table. For each artifact, the following attributes are displayed:

Attributes of Number Ranges Artifacts

Attribute	Description
Name	Displays name of the artifact
Minimum Value	Minimum value of the artifact should be greater than or equal to 0.
Maximum Value	Maximum value of the artifact should be less than 15 digit.
Next Value	Displays a value that can be used the next time you invoke this artifact.
Field Length	Displays the number of digits for the current value. If the value of **maximum value** attribute is 100 then the field length should be greater than 2, if the value of **maximum value** attribute is 99999 then the field length should be greater than 4 and so on. If the value of this attribute is 4 and the value of **current value** attribute is 7, then the number range is reflected as 0007 and not just 7. This refers to the concept of padding '0's to the value. If the value of this attribute is 0 then the value of **current value** attribute is flashed as it is with no padding on the number range. The maximum value allowed for this attribute is 99.
Rotate	If this attribute is set and the number range reaches specified **maximum value**, then the **current value** resets to specified **minimum value**.
Deployed By	Displays the user id of the user who deployed the artifact.
Deployed On	Displays the time when the artifact was deployed.

To sort and filter the content of the table, choose *Table Settings* (). On the subsequent screen, you can define how the table entries are to be sorted (by specifying an attribute and whether the entries are to be sorted for that attribute in ascending or descending order). You can also filter the table entries for certain attributes.

Actions

To create/deploy a new artifact, choose *Add*.

To edit and redeploy an existing artifact, select the artifact in the table and choose *Edit*.

To undeploy an artifact, select the artifact in the table and choose *Undeploy*.

7 Security Artifact Renewal

Security artifacts like certificates or passwords (for example) are subject to a specific lifecycle, in other words, they expire in certain time periods. To make sure that the operation of a scenario (using security artifacts) can be continued without any downtime, the process to renew security artifacts has to be performed in a coordinated way by the administrators of the involved components.

For the different use cases specific security artifact renewal processes have been defined.

> **Note**
>
> Note the following with regard to terminology:
>
> - The terms *client* and *server* are preferably used in the context of the certificate-based authentication option for HTTPS-based communication (transport level security). The background of this is that in order to set up a mutual authentication (that comes with this option), certificates for both roles, *client* and *server*, are required. When a message is sent from a sender (which has the role of a client) to a receiver (which has the role of a server), authentication steps are executed both to check if the server is a trusted partner and if the client is allowed to call the server.
> - In the context of message level security, the terms *sender* and *receiver* are preferably used in order to simplify things. These use cases typically require the following kinds of certificates or keys:
>   - Keys owned by a *sender* to either encrypt or sign the content of a message
>   - Keys owned by a *receiver* to either verify or decrypt the content of a message

Related Information

Basic Security Artifact Renewal Processes [page 146]
Renewal of Keys Provided by SAP [page 201]

7.1 Basic Security Artifact Renewal Processes

> **Note**
>
> These topics describe separate processes for renewing the various keys that are involved in an integration scenario using SAP Cloud Platform Integration. The topics cover situations where a key is owned either by the tenant administrator or by the administrator of the sender/receiver system connected to the tenant.
>
> To keep things simple, the individual topics describe idealized situations where a key pair is used in a **single** step or only in **one** communication channel. In real-life situations, however, a key pair is typically used in several integration flows, integration flow steps, and communication channels. The tenant administrator

needs to know all the steps and channels where the key pair is used to be able to correctly define the key renewal process.

Related Information

Use Cases [page 147]
Involved Roles [page 149]
Security Artifact Renewal for Transport Level Security [page 150]
Security Artifact Renewal for Message Encryption/Signature [page 165]

7.1.1 Use Cases

The following tables provides a list of all use cases and links to the corresponding renewal procedures.

Transport Level Security Use Cases

Security Artifact Renewal Use Cases (Transport Level)

Protocol	Authentication Option	Direction	Renewal Procedure
HTTPS	Certificate-Based	Outbound	Renewal of Tenant Client Root Certificate (CA) [page 151]
			Renewal of the Tenant Client Certificate [page 153]
			Renewal of Receiver Back-End Server Certificate [page 155] (also applicable in case a SuccessFactors receiver channel is used)
		Inbound	Renewal of Load Balancer Server Certificate [page 156]
			Renewal of Sender Back-End Client Certificate [page 157]
	Basic	Outbound	Renewal of User and Password [page 159] (also applicable in case a SuccessFactors receiver channel is used)
			Renewal of Password Only [page 160] (also applicable in case a SuccessFactors receiver channel is used)
		Inbound	Renewal of User and Password [page 161]
			Renewal of Password Only [page 161]
SFTP	Certificate-Based	Tenant pulls data from SFTP server	Renewal of the SFTP Server Key [page 162]
			Renewal of the SFTP Client Key (on Tenant) [page 163]
		Tenant pushes data to SFTP server	Renewal of User on SFTP Server [page 164]

Message Level Security Use Cases

Security Artifact Renewal Use Cases (Message Level Security)

Standard	Protection Method	Renewal Procedure
CMS/PKCS#7	Signer (Tenant signs outbound message)	Renewal of Keys for CMS/PKCS#7 Signer - Outbound [page 165] (key pair renewed on tenant)
	Verifier (Tenant verifies inbound message)	Renewal of Keys for CMS/PKCS#7 Verifier - Inbound [page 168] (key pair renewed by sender)
	Encryptor (Tenant encrypts outbound message)	Renewal of Keys for CMS/PKCS#7 Encryptor - Outbound [page 170] (key pair renewed by receiver)
	Decryptor (Tenant decrypts outbound message)	Renewal of Keys for CMS/PKCS#7 Decryptor - Inbound [page 173] (key pair renewed on tenant)
OpenPGP	Encryption key (Tenant encrypts outbound message)	Renewal of OpenPGP Encryption Key - Outbound [page 175] (encryption key renewed by receiver)
	Encryption key (Tenant decrypts inbound message)	Renewal of OpenPGP Encryption Key - Inbound [page 178] (encryption key renewed on tenant)
	Signer Key (Tenant decrypts outbound message)	Renewal of OpenPGP Signer Key - Outbound [page 179] (signer key renewed on tenant)
	Signer key (Tenant verifies inbound message)	Renewal of OpenPGP Signer Key - Inbound [page 181] (signer key renewed by sender)
XML Digital Signature	Signer (Tenant signs outbound message)	Renewal of Keys for XML Digital Signature Signer - Outbound [page 182]
	Verifier (Tenant verifies inbound message)	Renewal of Keys for XML Digital Signature Verifier - Inbound [page 184]
WS-Security	Signer (Tenant signs outbound request message)	Security Artifact Renewal for WS-Security (Tenant Signs Outbound Request) [page 195]
	Signer (Tenant signs response message (to an inbound request))	Security Artifact Renewal for WS-Security (Tenant Signs Inbound Response) [page 192]
	Verifier (Tenant verifies inbound request message)	Security Artifact Renewal for WS-Security (Tenant Verifies Inbound Request) [page 189]

Standard	Protection Method	Renewal Procedure
	Verifier (Tenant verifies response message (to an outbound request))	Security Artifact Renewal for WS-Security (Tenant Verifies Outbound Response) [page 198]
	Encryptor (Tenant encrypts outbound request message)	Security Artifact Renewal for WS-Security (Tenant Encrypts Outbound Request) [page 196]
	Encryptor (Tenant encrypts response message (to an inbound request))	Security Artifact Renewal for WS-Security (Tenant Encrypts Inbound Response) [page 194]
	Decryptor (Tenant decrypts inbound request message)	Security Artifact Renewal for WS-Security (Tenant Decrypts Inbound Request) [page 190]
	Decryptor (Tenant decrypts response message(to an outbound request))	Security Artifact Renewal for WS-Security (Tenant Decrypts Outbound Response) [page 200]

7.1.2 Involved Roles

The security artifact renewal process requires that different persons perform a sequence of steps in a coordinated way on each side of the communication. The exact sequence depends on the kind of security material which is renewed and on the use case.

Roles in the Security Artifact Renewal Process

Role	Tasks
Sender/receiver administrator (at customer side)	Updates the security artifacts owned by the sender/receiver back-end system (for example, the keystore).
Integration developer	Updates the integration flow in certain use cases. It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.
Tenant administrator	Updates the security artifacts of the tenant (relevant for outbound communication). It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Role	Tasks
SAP administrator	SAP administrator -Performs further settings at SAP, for example, the update of the security artifacts of the load balancer (relevant for inbound communication).

7.1.3 Security Artifact Renewal for Transport Level Security

7.1.3.1 Security Artifact Renewal for HTTPS-Based Communication

Using HTTPS, you can specify two different authentication options: certificate-based authentication and basic authentication. These options imply a different set of security artifacts for which specific renewal processes exist.

Certificate-based authentication (through HTTPS) involves the usage of X.509 SSL certificates both at client and server side. These certificates expire at a specified point in time. To ensure operation of scenarios using this communication type without any downtime requires the coordinated renewal of certificates both at client and server side.

Basic authentication uses credentials to allow the identification of trusted communication partners. Credentials are composed of user and password and, in case the SuccessFactors connector is involved, additional attributes. In addition to credentials, basic authentication also uses a one-way SSL connection which requires server certificates. Therefore, security artifact update involves both the update of credentials and of the involved certificates.

7.1.3.1.1 Certificate-Based Authentication (Outbound)

Certificate-based outbound authentication involves the usage of X.509 SSL certificates both at client and server side. For outbound communication, the keystore of the tenant is involved.

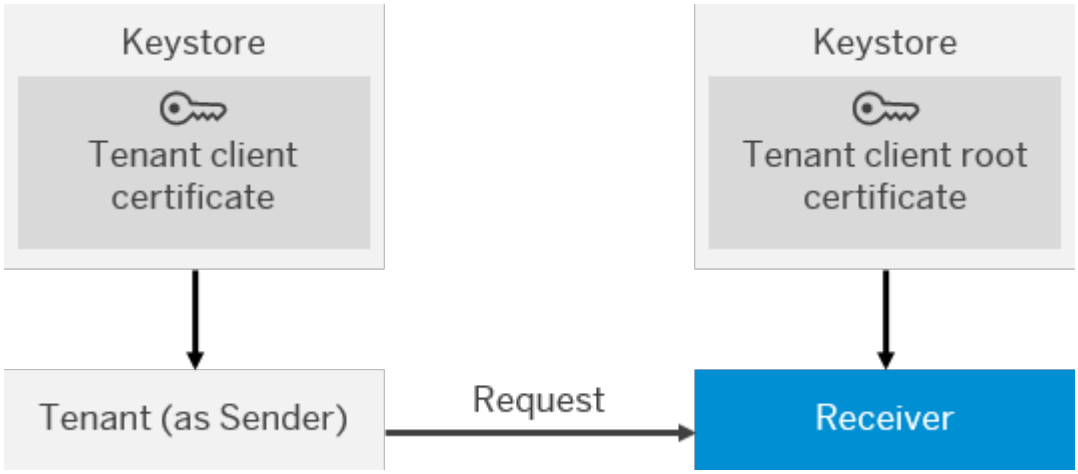
Related Information

Renewal of the Tenant Client Certificate [page 153]
Renewal of Receiver Back-End Server Certificate [page 155]

7.1.3.1.1.1 Renewal of Tenant Client Root Certificate (CA)

In this use case, the tenant client certificate is exchanged by a new one signed from a different certification authority (CA).

The following figure illustrates the communication path that is relevant for this use case.



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Receiver Accepts Different Certificates at the same Time

Certificate renewal has to be performed in the following sequence:

1. Tenant administrator: Creates new certificate with a new key pair and gets the certificate signed by another CA than the old one.
2. Tenant administrator: Provides receiver administrator with the new certificate and root certificate (the latter one because the CA had changed).
3. Receiver administrator: Configures receiver (server) that way that the old and the new client certificate are accepted by the server.
   Because the receiver administrator also has received a new **root certificate**, this root certificate also has to be imported into the server keystore.
4. Receiver administrator: Informs the tenant administrator that the receiver system (server) now accepts both the old and the new client certificate.
5. Tenant administrator: Exchanges the old key pair/certificate with the new key pair/certificate - keeping the old alias in the tenant client keystore.
   This is done by importing the new signed certificate into the tenant client keystore.
6. Tenant administrator: Informs the integration developer that the integration flow can be restarted.

7. Integration developer: Restarts the integration flow which sends data via HTTPS to the receiver system. This is necessary because the SSL socket caches the keystore for 24 hours.
   If your tenant cluster contains multiple runtime nodes, make sure that you restart your integration flow on all nodes. If you start the integration flow on only one runtime node, message processing might fail on the other nodes.
8. Tenant administrator: Informs the receiver administrator that a new client certificate (signed by another CA than the old one) is now used.
9. Receiver administrator: Removes the old client certificate and also the old root certificate (assumed that it is not longer used in any other communication).

Let us assume, the customer landscape is composed as described under *Connecting a Customer System to Cloud Integration*, section *Technical Landscape for On Premise-On Demand Integration*. In that case, SAP Web Dispatcher is used to receive incoming calls from the SAP Cloud. SAP Web Dispatcher (as reverse proxy) is the entry point for HTTPS requests into the customer system landscape. The configuration of the receiver (server) as indicated in step 2 in the list above comprises the following tasks for that example case:

- Make sure that the reverse proxy trusts the new CA. A restart is required to finalize the related configuration steps.
- Map the new certificate in AS ABAP back-end for authentication purpose
- Edit the new CA in Web Dispatcher farm.
  This step is performed by SAP IT.
- Upload the new CA in workcenter under *Edit Certificate Trust List*.
- Update the communication arrangements credentials such way that the new certificate is mapped to the inbound technical user.

Receiver does not Accept Different Certificates at the same Time

Certificate renewal has to be performed in the following sequence:

1. Tenant administrator: Creates new certificate with a new key pair and gets the certificate signed by another CA than the old one.
2. Tenant administrator and Receiver administrator: Aggree on a downtime window.
3. At start of the downtim window, the tenant administrator informs the integration developer that the integration flow which uses the client certificate for outbound HTTPS communication has to be stopped.
4. Integration developer: Stops the integration flow.
5. Integration developer: Informs the tenant administrator that the integration flow has been stopped.
6. Tenant administrator: Exchanges key pair/certificate in the keystore keeping the old alias.
   This is done by importing the new signed certificate into the tenant client keystore.
7. Tenant administrator: Provides receiver administrator with the new certificate and the root (CA) certificate. The tenant administrator informs the receiver administrator that the HTTPS client has been stopped.
8. Receiver administrator: Exchanges the certificate and imports the new root (CA) certificate into the truststore.
9. Receiver administrator: Informs the tenant administrator that the certificate has been exchanged.
10. Tenant administrator: Informs the integration developer that the integration flow can be restarted.
11. Integration developer: Restarts the integration flow which sends data via HTTPS to the receiver system. This is necessary because the SSL socket caches the keystore for 24 hours.

If your tenant cluster contains multiple runtime nodes, make sure that you restart your integration flow on all nodes. If you start the integration flow on only one runtime node, message processing might fail on the other nodes.

> **Note**
>
> It is not expected that this case occurs very often.

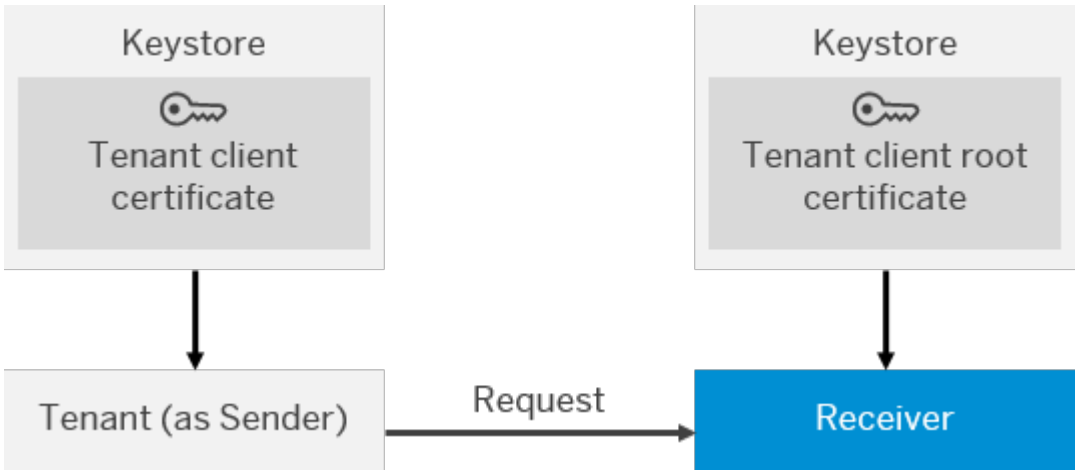
Related Information

Involved Roles [page 149]

7.1.3.1.1.2 Renewal of the Tenant Client Certificate

In this use case, the tenant client certificate has to be renewed. In the renewal process, the tenant administrator (managing the tenant cluster) and the integration developer (managing the integration flow deployed on the tenant) collaborate with the administrator of the receiver back-end system.

The following figure illustrates the communication path that is relevant for this use case.



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Receiver Accepts Different Certificates at the same Time

Certificate renewal has to be performed in the following sequence:

1. Tenant administrator: Creates new certificate with a new key pair and gets the certificate signed by a CA.
2. Tenant administrator: Provides receiver administrator with the new certificate and root certificate (if the CA had changed).
3. Receiver administrator: Configures receiver (server) that way that the old and the new client certificate are accepted by the server.
   In case the receiver administrator also has received a new root certificate, this root certificate also has to be imported into the server keystore.
4. Receiver administrator: Informs the tenant administrator that the receiver system (server) now accepts both the old and the new client certificate.
5. Tenant administrator: Exchanges the old key pair/certificate with the new key pair/certificate - keeping the old alias in the tenant client keystore.
   This is done by importing the new signed certificate into the tenant client keystore.
6. Tenant administrator: Informs the integration developer that the integration flow can be restarted.
7. Integration developer: Restarts the integration flow which sends data via HTTPS to the receiver system.
   This is necessary because the SSL socket caches the keystore for 24 hours.
   If your tenant cluster contains multiple runtime nodes, make sure that you restart your integration flow on all nodes. If you start the integration flow on only one runtime node, message processing might fail on the other nodes.
8. Tenant administrator: Informs the receiver administrator that the new client certificate is now used.
9. Receiver administrator: Removes the old client certificate and (if required) also the old root certificate (assumed that it is not longer used in any other communication).

Receiver does not Accept Different Certificates at the same Time

Certificate renewal has to be performed in the following sequence:

1. Tenant administrator: Creates new certificate with a new key pair and gets the certificate signed by a CA.
2. Tenant administrator and Receiver administrator: Aggree on a downtime window.
3. At start of the downtim window, the tenant administrator informs the integration developer that the integration flow which uses the client certificate for outbound HTTPS communication has to be stopped.
4. Integration developer: Stops the integration flow.
5. Integration developer: Informs the tenant administrator that the integration flow has been stopped.
6. Tenant administrator: Exchanges key pair/certificate in the keystore keeping the old alias.
   This is done by importing the new signed certificate into the tenant client keystore.
7. Tenant administrator: Provides receiver administrator with the new certificate and the root (CA) certificate if the latter has been changed.
   The tenant administrator informs the receiver administrator that the HTTPS client has been stopped.
8. Receiver administrator: Exchanges the certificate and imports the new root (CA) certificate into the truststore (in case a new root certificate has been received).
9. Receiver administrator: Informs the tenant administrator that the certificate has been exchanged.
10. Tenant administrator: Informs the integration developer that the integration flow can be restarted.
11. Integration developer: Restarts the integration flow which sends data via HTTPS to the receiver system.

This is necessary because the SSL socket caches the keystore for 24 hours.
If your tenant cluster contains multiple runtime nodes, make sure that you restart your integration flow on all nodes. If you start the integration flow on only one runtime node, message processing might fail on the other nodes.

> **Note**
>
> It is not expected that this case occurs very often.

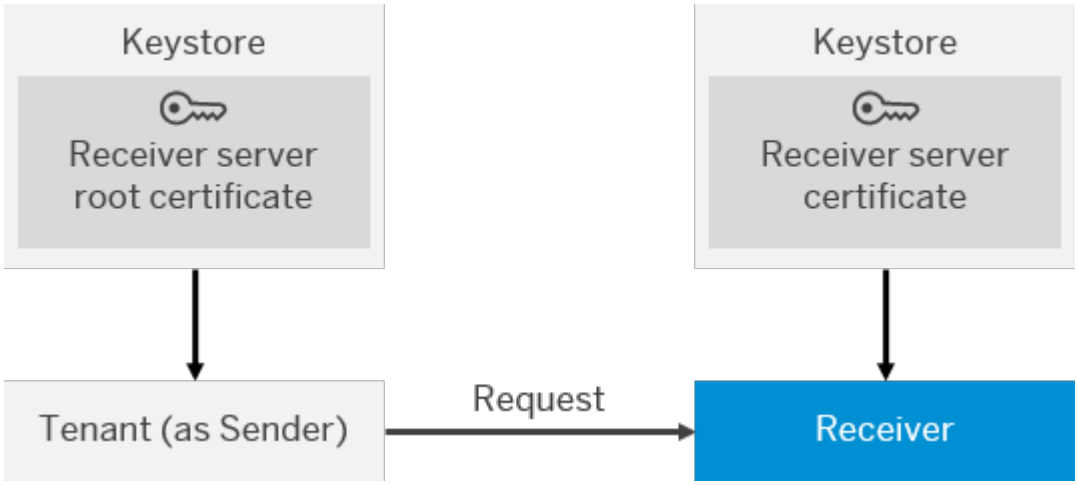
Related Information

Involved Roles [page 149]

7.1.3.1.1.3 Renewal of Receiver Back-End Server Certificate

In this use case, the server certificate (of the receiver) has to be renewed.

The following figure illustrates the communication path that is relevant for this use case.



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Certificate renewal has to be performed in the following sequence:

> **Note**
>
> This process is also applicable in case a SuccessFactors adapter is used in the receiver channel.

> **Note**
>
> It is assumed here that the exchange of the server certificate does not cause any downtime of the server. The load balancer, AS ABAP (in case the receiver is an SAP system based on AS ABAP) and also the SAP JEE Engine support this.

1. Receiver administrator: Ceates key pair/certificate for the receiver (server) and uses a different CA certificate to sign the server certificate.
2. Receiver administrator: Provides the tenant administrator with the server root certificate (of the CA).
3. Tenant administrator: Imports the root certificate into the tenant client keystore (of the tenant).
4. Tenant administrator: Restarts all integration flows which are sending via HTTPS data to the receiver system.
   This is required because the SSL socket caches the keystore for 24 hours.
5. Tenant administrator: Informs receiver administrator that root certificate has been added.
6. Receiver administrator: Exchanges the key pair/certificate in the receiver system.
7. Receiver administrator: Informs the tenant administrator that the old root certificate can be removed.
8. Tenant administrator: Deletes the old root certificate from the tenant client keystore.

Related Information

Involved Roles [page 149]

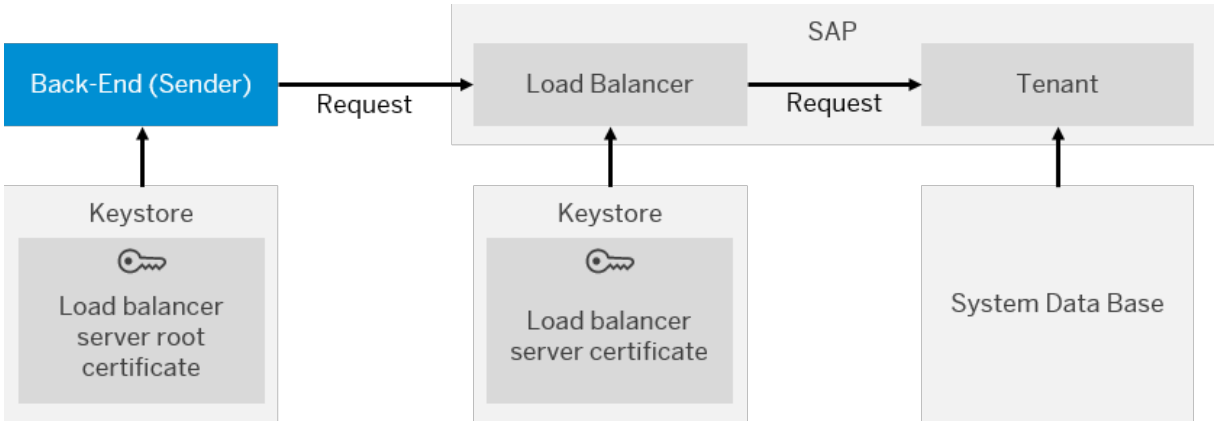
7.1.3.1.2 Certificate-Based Authentication (Inbound)

Certificate-based outbound authentication involves the usage of X.509 SSL certificates both at client and server side.

7.1.3.1.2.1 Renewal of Load Balancer Server Certificate

In this use case, the load balancer server certificate at SAP has to be renewed. In the renewal process, the load balancer administrator (at SAP) and the sender back-end administrator (at the customer side) collaborate with each other.

The following figure illustrates the communication path that is relevant for this use case.



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Certificate renewal has to be performed in the following sequence.

1. Load balancer administrator: Creates new key pair/certificate with new CA root certificate.
2. Load balancer administrator: Informs tenant administrator that virtual server certificate will be exchanged at a certain point in time and forwards new root (CA) certificate.
3. Tenant administrator: Informs sender administrator and forwards the new root certificate to the sender.
4. Sender administrator: Adds the new root certificate to the truststore of the sender back-end (HTTPS client).
5. Load balancer administrator: Exchanges the load balancer virtual server key pair/certificate at the specified point in time.
6. Sender administrator: Can now remove the old root certificate form the truststore of the sender back-end (HTTPS client) after the specified point in time has passed.

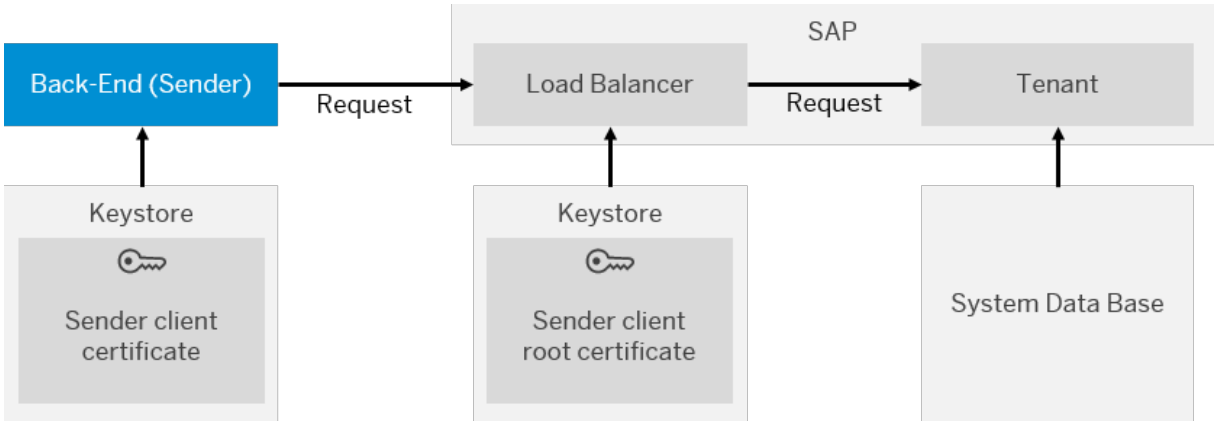
Related Information

Involved Roles [page 149]

7.1.3.1.2.2 Renewal of Sender Back-End Client Certificate

In this use case, the client certificate (of the sender) has to be renewed.

The following figure illustrates the communication path that is relevant for this use case.



Note

It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Certificate revewal has to be performed in the following sequence.

1. Sender administrator: Creates new key pair/client certificate (also the root certificate (CA) may be changed).
2. Sender administrator: Provides tenant administrator with the certificate and the root certificate. Sender administrator has to make sure that the client certificate in the sender keystore is signed by one CA that is listed in Load Balancer Root Certificates Supported by SAP [page 225].
3. After the CA certificate was added to the truststore of the virtual server, tenant administrator forwards the certificate to the integration developer and asks him to add the subject and issuer DN of the certificate to the authorization interceptor of the inbound channel corresponding to the sender system.
4. Integration developer: Adds the DNs of the new certificate to the authorization interceptor in the integration flow and redeploys the integration flow.
5. Integration developer: Informs the tenant administrator that authorization interceptor has been configured with the new certificate.
6. Tenant administrator: Informs the sender administrator that he now can sent messages with the new client certificate.
7. Sender administrator: Configures sender system that way that it sends HTTPS messages with the new client certificate.
8. Sender administrator: Informs tenant administrator that sender system is using now the new client certificate for the HTTPS communication.
9. Tenant administrator: Informs integration developer that he can remove the DNs of the old certificate from the authorization interceptor in the integration flow.
10. Integration developer: Removes the DNs of the old certificate from the authorization interceptor in the integration flow and redeploys the integration flow.

Note

Steps 4-6 and 10-11 are only necessary if the subject DN or issuer DN of the certificate has been changed.

Related Information

Involved Roles [page 149]

7.1.3.1.3 Basic Authentication (Outbound)

Basic authentication uses credentials to allow the identification of trusted communication partners. Credentials are composed of user and password and, in case the SuccessFactors connector is involved, additional attributes. In addition to credentials, basic authentication also uses a one-way SSL connection which requires server certificates. Therefore, security artifact update involves both the update of credentials and of the involved certificates.

7.1.3.1.3.1 Renewal of User and Password

In this use case, the user (through which the tenant calls the receiver system) is replaced by a new user (and password) in the receiver system.

> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Security artifact renewal has to be performed in the following sequence:

1. Receiver administrator: Creates a new user (`user1`) and assigns authorization roles to the user.
   After this step has been performed, two users are configured on the receiver system for a certain HTTPS communication: the old user (`user0`) and the new one (`user1`).
2. Receiver administrator: Informs the tenant administrator that he wants to exchange the old user (`user0`) with a new user (`user1`).
   The new user also should have a new password.
3. Tenant administrator: Starts the Integration Operations user interface, opens the related *User Credentials* artifact (specified for `user0` and the communication path with the receiver system) and exchanges the old user/password with the new user/password.
4. Tenant administrator: Restarts the corresponding integration flow(s) using the Integration Operations user interface.
5. Tenant administrator: Informs the receiver administrator that user/password has been exchanged.
6. Receiver administrator: Removes the old user.

In case a **SuccessFactors** receiver channel is used, note the following, slightly adapted sequence of steps:

1. Success Factors administrator (receiver administrator): Creates a new user/password and assigns the adequate authorizations to the user for the new company ID.

2. Success Factors administrator: Informs the tenant administrator that a new user/password for new company ID has been created and that the old user/password/company ID will be no longer valid from a certain point in time.
3. Tenant administrator: Changes the user/password/company ID in the Integration Operations user interface for the existing credentials (User Credentials artifact).
4. Success Factors administrator: Removes old user/password/company ID from Success Factors system after the specified point in time has been reached.

Related Information

Involved Roles [page 149]

7.1.3.1.3.2 Renewal of Password Only

In this use case, the password of the user through which the tenant calls the receiver system is replaced in the receiver system.

> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

To exchange the password of a user without any downtime, the receiver administrator has to create an intermediate user as described for the use case *Renewal of User and Password* (without deleting the old user).

1. Receiver administrator: Creates a new **intermediate user** (`user1`) and assigns authorization roles to the user.
   After this step has been performed, two users are configured on the receiver system for a certain HTTPS communication: the old user (`user0`) and the new one (`user1`).
2. Receiver administrator: Informs the tenant administrator that he wants to change the password of a certain user used for a certain HTTPS communication and that he has created an intermediate user (`user1`) with a certain password.
3. Tenant administrator: Starts the Integration Operations user interface, opens the related *User Credentials* artifact (specified for `user0` and the communication path with the receiver system) and exchanges the old user0/password with the new user1/password.
4. Tenant administrator: Informs the receiver administrator that the client now uses the intermediate user (`user1`).
5. Receiver administrator: Changes the password of the original user (`user0`).
6. Receiver administrator: Informs the tenant administrator that the password of the original user (`user0`) has been changed.
7. Tenant administrator: Starts the Integration Operations user interface, opens the related *User Credentials* artifact (now containing the credentials of `user1`) and exchanges the credentials (user/password) of the

intermediate user (`user1`) with the credentials (user name and new password) of the original user (`user0`).

8. Tenant administrator: Informs the receiver administrator that user password has been changed.
9. Receiver administrator: Removes the intermediate user.

> **Note**
>
> In case the receiver administrator does not accept this complicated procedure, a simplified procedure might be adopted that way that the tenant administrator just changes the password as soon as he notices that the connection to the receiver system fails due to a wrong password.

> **Note**
>
> The same procedure is applicable in case a SuccessFactors receiver channel is used.

Related Information

Involved Roles [page 149]

7.1.3.1.4 Basic Authentication (Inbound)

7.1.3.1.4.1 Renewal of User and Password

In this use case, the user (through which a sender calls the tenant) is replaced by a new user (and password) on the SAP cloud platform.

Security artifact renewal has to be performed in the following sequence:

1. SAP: Informs the sender administrator that the sender back-end system should use new user/password for communication with the tenant.
2. Sender administrator: Changes the user and password in the HTTPS sender client (sender back-end).
3. Sender administrator: Informs SAP that user has been changed in the sender client.

7.1.3.1.4.2 Renewal of Password Only

In this use case, the password of the user (through which the sender system is supposed to call the tenant) is replaced by a new password on the SAP cloud platform.

To exchange the password of a user without any downtime, SAP has to create an intermediate user as described for the use case *Renewal of User and Password* (without deleting the old user).

Security artifact renewal has to be performed in the following sequence:

1. SAP: Informs the sender administrator that he wants to change the password of a certain user used for HTTPS communication with the tenant and that he has created an **intermediate user** (`user1`) and password.
2. Sender administrator: Exchanges the old user/password (`user0`) with the intermediate user/password (`user1`) in the HTTPS sender client (back-end system).
3. Sender administrator: Informs SAP that the sender client now uses the intermediate user (`user1`).
4. SAP: Informs the sender administrator that the password of the original user (`user0`) has been changed.
5. Sender administrator: Exchanges the user/password of the intermediate user (`user1`) with the original user (`user0`) (and with the new password).
6. Sender administrator: Informs SAP that user and password has been changed.

7.1.3.2 Security Artifact Renewal for SFTP-Based Communication

Using SSH File Transfer Protocol (SFTP), the basic set up is that an SFTP client is connected to an SFTP server from which the client pulls data or to which the client pushes data. Secure SFTP communication is enabled by the usage of public/private key pairs as well as a trust relationship between client and server implemented by known_hosts files.

In scenarios using SFTP, the tenant is always an SFTP client either pushing files to the SFTP server or pulling files from it.

Where the SFTP is hosted, depends on the scenario.

In SFTP security artifact renewal processes, the following roles are involved:

- SFTP server administrator
- Tenant administrator (is always the *SFTP client administrator*)
- Integration developer

The following security artifacts can be subject to change and underly a renewal process:

- Public/private key pair (of either SFTP server or tenant)
- User who either pushes files to SFTP server or pulls files from it

7.1.3.2.1 Renewal of the SFTP Server Key

In this use case, an SSH key pair is renewed on the SFTP server.

> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Security artifact renewal has to be performed in the following sequence:

1. SFTP server administrator: Creates new server key pair.
2. SFTP server administrator: Provides tenant administrator with the public key and informs him that the key will be exchanged on the SFTP server at a certain point in time.
3. Tenant administrator: Adds the new public key to the *known hosts* file.
   After that step has been performed, the client (tenant) trusts the server either he has the old or new key.
4. Tenant administrator: Removes the old public key entry form the *known hosts* file after the aggreed point in time.

Related Information

Involved Roles [page 149]

7.1.3.2.2 Renewal of the SFTP Client Key (on Tenant)

In this use case, an SSH key pair is renewed on the SFTP client (tenant).

> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Security artifact renewal has to be performed in the following sequence:

1. Tenant administrator: Generates new key pair for the SFTP client.
2. Tenant administrator: Exports the public key from the keystore and provides the SFTP server administrator with the public key.
3. SFTP server administrator: Imports the public key (provided by the tenant administrator) into the SFTP server truststore.
4. SFTP server administrator: Informs tenant administrator that public key has been imported into the truststore of the SFTP server.
5. Tenant administrator: Exchanges in the keystore the old key pair with the new one.

Related Information

Involved Roles [page 149]

7.1.3.2.3 Renewal of User on SFTP Server

Files are stored on the SFTP server in directories referred to as mailboxes. For each mailbox, a user is specified to control access to the data. In this use case, the mailbox user on the SFTP server is changed.

> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

There are two sub use cases, depending on whether the tenant pulls data from or pushes data to the SFTP server.

Tenant Pulls Data from SFTP Server

User renewal has to be performed in the following sequence:

1. SFTP server administrator: Creates a new user on the SFTP server with all relevant configurations (for example, mailbox).
2. SFTP server administrator: Informs the tenant administrator that an old user shall be exchanged by a new one and that the new user already exists on the SFTP server.
3. Tenant administrator: Informs integration developer that he should exchange the old SFTP user with the new one (in the corresponding integration flow).
4. Integration developer: Exchanges the old SFTP user with the new one in the integration flow.
5. Integration developer: Informs the tenant administrator that user has been exchanged.
6. Tenant administrator: Informs SFTP server administrator that the user has been exchanged.
7. SFTP server administrator: Makes sure that all data of the old user has been fetched by the tenant. If this is not the case he transfers the relevant data into the mailbox of the new user.

Tenant Pushes Data to SFTP Server

User renewal has to be performed in the following sequence:

1. SFTP server administrator: Creates a new user on the SFTP server with all relevant configurations (for example, mailbox).
2. SFTP server administrator: Informs the tenant administrator that an old user shall be exchanged by a new one and that the new user already exists on the SFTP server.
3. Tenant administrator: Informs the integration developer that he should exchange the old SFTP user with the new one (in the corresponding integration flow).
4. Integration developer: Exchanges the old SFTP user with the new one in the integration flow.
5. Integration developer: Informs the tenant administrator that user has been exchanged.
6. Tenant administrator: Informs the SFTP server administrator that the user has been exchanged.

7. If a pulling component relies on the data, the SFTP server administrator makes sure that the poller has read all data from the mailbox of the old user. If this is not the case, the SFTP server administrator transfers the data into the mailbox of the new user.

Related Information

Involved Roles [page 149]

7.1.4 Security Artifact Renewal for Message Encryption/ Signature

Related Information

Message-Level Security [page 227]

7.1.4.1 Security Artifact Renewal for PKCS#7/CMS

Related Information

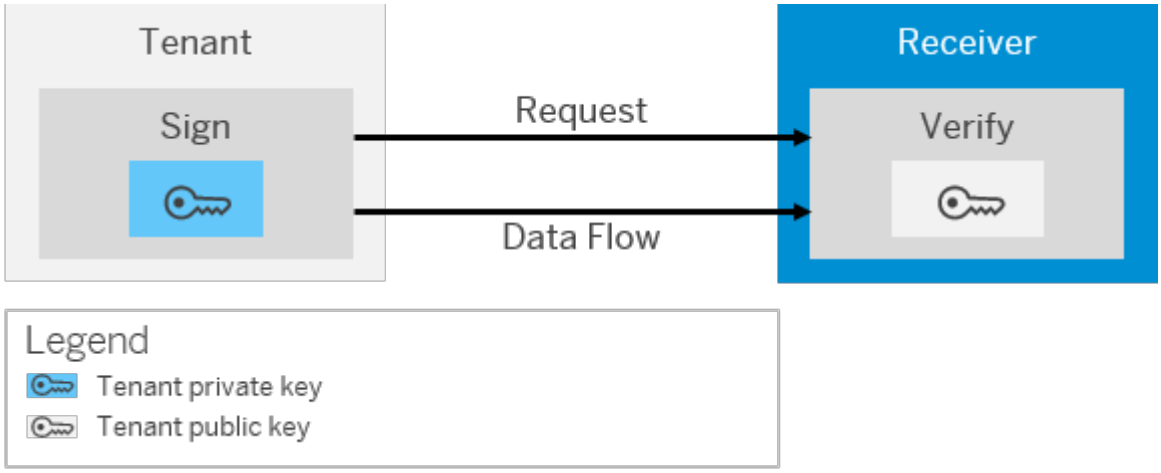
How PKCS#7 Works [page 230]

7.1.4.1.1 Renewal of Keys for CMS/PKCS#7 Signer - Outbound

This use case covers all situations where private keys (used by the tenant to sign outbound messages) are changed. The renewal process ensures that the related public verification key is changed at the receiver side that way that no downtime is required.

The signer (when configured to use the CMS/PKCS#7 standard) uses one or more private keys to sign a single payload. These private keys are provided in the outbound keystore of the tenant. The resulting signed data object can contain several signatures from different private keys. To locate the different private keys in the keystore, aliases can be specified in the corresponding integration flow signer step.

The following figure illustrates the communication path that is relevant for this use case.



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

The renewal process depends on whether the receiver system can verify signed data with different public keys at one point in time.

Receiver is able to Verify Payloads Signed by the old Key and Payloads Signed by the new Key at the Same Time

This renewal process implies the following sequence of steps.

1. Tenant administrator: Creates a new key pair.
2. Tenant administrator: Provides the new certificate to the receiver administrator.
3. Receiver administrator: Configures the receiver system that way that it is able to verify payloads signed by the old key or payloads signed by the new key.
4. Receiver administrator: Informs the tenant administrator that the receiver system is able to verify payloads signed by the old key or payloads signed by the new key.
5. Tenant administrator: Exchanges the old key pair with the new key pair, keeping the old alias.
   From now on outbound messages are signed with the new key.
6. Tenant administrator: Informs the receiver administrator that the key pair has been exchanged.
7. Receiver administrator: Removes the old key pair.
   From now on, the receiver system can only verify payloads signed by the new key.

Receiver is only able to Verify Payloads Signed by the Same Key at one Point in Time

This renewal process requires cooperation of tenant administrator, integration developer and receiver administrator.

> **Note**
>
> Is is assumed that the receiver system can manage CMS/PKCS#7-signed data containing several signatures. This should be the case because the specification requires it.

1. Tenant administrator: Creates a new key pair.
2. Tenant administrator: Adds the new key pair to the keystore with a new alias.
3. Tenant administrator: Informs the integration developer.
4. Integration developer: Changes the integration flow.
   In particular, the integration developer adds the new alias to the Signer Parameters table of the CMS/PKCS7 Signer step. All other parameters like *Signature Algorithm*, *Include Certificates*, *Include Signing Time* have to be the same as specified for the entry with the old alias.
   From now on, the tenant sends signed data containing two signatures both from the old and the new key.
5. Tenant administrator: Provides the new certificate to the receiver administrator and informs him that the tenant is sending from now on signed data containing two signatures, a signature of the old key and a signature of the new key.
6. Receiver administrator: Exchanges the key pair that way that the receiver system verifies from now on the signature of the new key.
7. Receiver administrator: Informs the tenant administrator that the receiver system verifies the signature of the new key.
8. Tenant administrator: Informs the integration developer that he can remove the old alias from the integration flow signer step.
9. Integration developer: Removes the old alias from the integration flow signer step.
   From now on, the tenant sends signed data with only one signature of the new key.
10. Integration developer: Informs the tenant administrator that the alias has been removed.
11. Tenant administrator: Removes the old key pair from the keystore.

Receiver can only Verify Payloads Signed by the Same Key at one Point in Time but Accepts PKCS7/CMS Data with two Signatures

This use case is supported as of release 1.7 of the Integration Designer.

This procedure is not applicalbe for system based on AS ABAP.

The following assumptions apply:

- The receiver system can handle CMS/PKCS7-signed data containing several signatures. Actually this should be the case because this is part of the specification. Note that systems based on AS ABAP do not support this.
- The PKCS7/CMS signer step contains two aliases, one for the current private key and one for the new private key.

If this is not the case, the integration developer has to be be asked to add a second alias.

1. Tenant administrator: Creates a new key pair/certificate.
2. Tenant administrator: Adds the new certificate to the keystore, taking into account that the alias is used which is specified in the PKCS7/CMS signer step for the new certificate.
   From now on the PKCS7/CMS data format contains two signatures.
3. Tenant administrator: Provides the receiver administrator with the new certificate.
4. Receiver administrator: Exchanges the old certificate with the new certificate and performs relevant configuration steps.
5. Receiver administrator: Informs the tenant administrator that certificate has been exchanged.
6. Tenant administrator: Removes the old certificate from the keystore.

Receiver can only Verify Payloads Signed by the Same Key at one Point in Time and Accepts only PKCS/CMS Data with Exaclty one Signature

This procedure is applicaple for systems based on AS ABAP.

> **Note**
>
> This procedure implies a downtime.

1. Tenant administrator: Creates new key pair/certificate.
2. Tenant administrator and receiver administrator: Aggree on a downtime.
3. Tenant administrator: Provides receiver admin with the certificate.
4. During the downtime:
   1. Tenant administrator: Exchanges key pair/certificate in keystore, keeping the old alias.
   2. Receiver administrator: Exchanges the certificate in receiver system.

Related Information

How PKCS#7 Works [page 230]
Involved Roles [page 149]

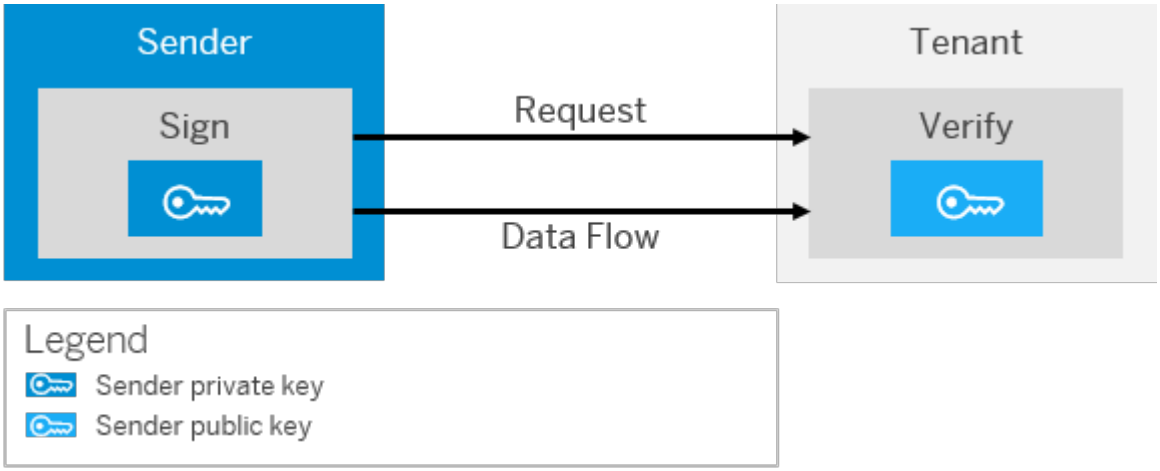
7.1.4.1.2 Renewal of Keys for CMS/PKCS#7 Verifier - Inbound

This use case covers all situations where private keys used by a sender to sign messages sent to the tenant (in our terminology: inbound messages) are changed. The renewal process ensures that the related public verification key is changed at the tenant side that way that no downtime is required.

The verifier (specified in the integration flow to use the CMS/PKCS#7 standard) uses a public key to verifiy a payload signed by the sender. This public key has been imported into the tenant keystore as X509 certificate

during the onboarding process. The verifier uses an alias configured in the corresponding integration flow step to locate the public key in the keystore. The renewal process depends on whether the sender system can send signed data with signatures from several keys. The CMS/PKCS#7 specification does allow this.

The following figure illustrates the communication path that is relevant for this use case.



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Sender is able to Send Payload Signed by Old and New Key

1. Sender administrator: Creates a new key pair.
2. Sender administrator: Configures the sender system that from now on it sends signed data with two signatures, from the old and new key.
3. Sender administrator: Provides new public key (certificate) to the tenant administrator.
4. Tenant administrator: Exchanges the old public key with the new one in the keystore, keeping the old alias. (From now on the avatar verifies hte signature of the new key.)
5. Tenant administrator: Informs the sender administrator that the key has been exchanged.
6. Sender administrator: Configures the sender system that way that a payload with one signature from the new key is being sent.
7. Sender administrator: Removes the old key pair.

Sender is Only able to Send Payload Signed by One Key

This procedure applies for Integration Designer as of Version 1.7.

For the renewal process it is assumed that the verifier integration flow step contains two aliases, one for the current certificate and one for the new certificate.

If the alias for the new certificate does not yet exist, the integration developer must add such an alias.

1. Sender administrator: Creates a new key pair.
2. Sender administrator: Provides the new public key to the tenant administrator.
3. Tenant administrator: Adds the new public key (certificate) to the keystore, taking into account that the alias is used which is specified in the verifier step for the new certificate.
4. Tenant administrator: Informs the sender administrator that payloads signed with the new key can be sent.
5. Sender administrator: Configures the sender system that way that from now on it sends payloads signed with the new key.
6. Sender administrator: Removes the old key pair.
7. Sender administrator: Informs the tenant administrator about the fact that the old key pair has been removed.
8. Tenant administrator: Removes the old public key (certificate) from the keystore after a time period which corresponds to the guaranteed delivery time (to make sure that payloads signed with the old key are no longer in the SAP cloud system).

Related Information

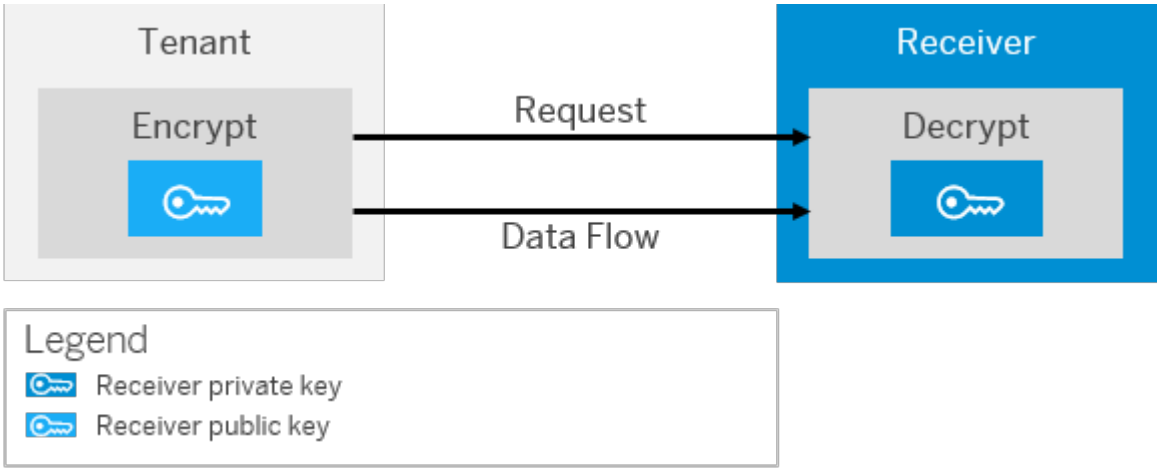
How PKCS#7 Works [page 230]
Involved Roles [page 149]

7.1.4.1.3 Renewal of Keys for CMS/PKCS#7 Encryptor - Outbound

This use case covers all situations where a private key used for message decryption is changed by a receiver. The renewal process ensures that the related public encryption key is changed at the tenant side that way that no downtime is required.

The encryptor (specified in the integration flow to use the CMS/PKCS#7 standard) uses one or serveral public keys to encrypt a payload. The resulting enveloped data then contains one or several recipient information elements corresponding to the public keys. These recipient information elements contain information about the certificates corresponding to the public keys (issuer DN and serial number of the certificate). The encryptor uses aliases configured in the integration flow step to loacte the certificates in the keystore.

The following figure illustrates the communication path that is relevant for this use case.



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Receiver is able to Decrypt Payloads Encrypted by the Old Key and Payloads Encrypted by the New Key at one Point in Time

1. Receiver administrator: Creates a new key pair.
2. Receiver administrator: Configures receiver system that way that it either can verify payloads encrytped with the old key or payloads encrypted with the new key.
3. Receiver administrator: Prrovides the new public to the tenant administrator.
4. Tenant administrator: Exchanges the public key (certificate) with the new public key, keeping the old alias. From now on, messages are being encrypted with the new key.
5. Tenant administrator: Informs the receiver administrator that certificate has been exchanged.
6. Receiver administrator: Removes the old key pair.

Receiver is only able to Decrypt Payloads Encrypted by the Same Key at one Point in Time

> **Note**
>
> It is assumed that receiver system can manage CMS/PKCS7 enveloped data containing several encryption recipients.

1. Receiver administrator: Creates a new key pair.

2. Receiver administrator: Provides the public key to the tenant administrator.
3. Tenant administrator: Adds the new public key to the keystore with a new alias.
4. Tenant administrator: Informs the integration developer that new certificate with new alias has been added for encryption usage.
5. Integration developer: Changes the integration flow.
   The integration developer adds the new alias to the *Receiver Public Key Alias* list of the CMS/PKCS7 *Encryptor* step.
   After this step has been performed, the tenant sends CMS/PKCS7 enveloped data containing two recipient information elements: from the old and from the new certificate.
6. Tenant administrator: Informs the receiver administrator that the tenant sends CMS/PKCS7 enveloped data with two recipient information elements: from the old and new certificate.
7. Receiver administrator: Exchanges the certificate that way tha the receiver system now uses the recipient information of the new certificate to decrypt the payload.
8. Receiver administrator: Removes the old key pair.
9. Receiver administrator: Informs the tenant administrator that the receiver now uses the recipient information of the new certificate to decrypt the payload.
10. Tenant administrator: Informs the integration developer that the old alias can be removed from the integration flow encryptor step.
11. Integration developer: Removes the old alias from the integration flow encryptor step.
    After this step has been performed, the sent CMS/PKCS7 enveloped data does only contain one recipient information of the new certificate.
12. Integration developer: Informs the tenant administrator that the alias has been removed.
13. Tenant administrator: Removes the old certificate from keystore.

Receiver can only Decrypt Payloads Encrypted by the Same Key at one Point in Time and Accepts PKCS7/CMS-Enveloped Data Containing Symmetric Keys Encrypted by Several Asymmetric Keys

This procedure is supported as of release 1.7 of the Integration Designer.

This procedure is applicable for systems based on AS ABAB.

The following assumptions apply:

- The receiver system can handle with CMS/PKCS7-enveloped data containing several encryption recipients. This should be the case because this is part of the specification.
- The PKCS7/CMS encryptor step contains two aliases, one for the current public key and one for the new public key. If this is not the case, the integration developer has to be asked to add a second alias.

1. Receiver administrator: Creates new key pair/certificate.
2. Receiver administrator: Provides new certificate to the tenant administrator.
3. Tenant administrator: Adds new certificate to the keystore taking into account that the alias is used which is specified in the PKCS7/CMS encryptor step for the new certificate.
   From now on the PKCS7/CMS enveloped data contains two encrpytions of the symmetric key.
4. Tenant administrator: Informs the receiver administrator that PKCS7/CMS-enveloped data are sent with two encryptions for the symmetric key.
5. Receiver administrator: Exchanges old key pair/certificate with the new one.

6. Receiver administrator: Informs the tenant administrator that the certificate has been exchanged.
7. Tenant administrator: Removes old certificate from the keystore.

Receiver can Only Decrypt Payloads Encrypted by the Same Key at one Point in Time and Does not Accept PKCS7/CMS-Enveloped Data Containing Symmetric Keys Encrypted by Several Asymmetric Keys

This procedure implies a downtime.

1. Receiver administrator: Creates new key pair/certificate.
2. Receiver administrator and tenant administrator: Aggree on a downtime.
3. Receiver administrator: Provides tenant administrator with certificate.
4. During the downtime:
    1. Receiver administrator: Exchanges the old key pair/certificate with the new one.
    2. Tenant administrator: Exchanges the old certificate with the new one, keeping the old alias

Related Information

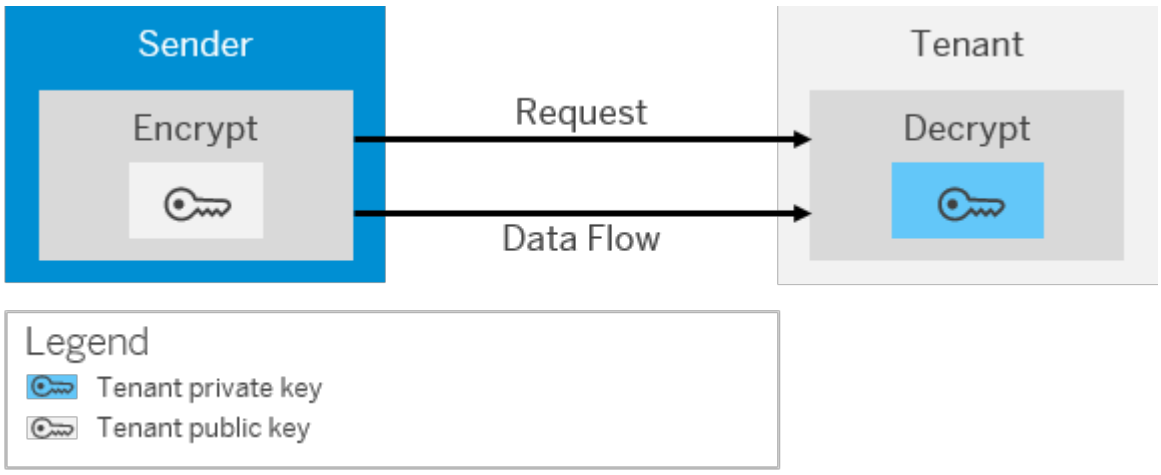
How PKCS#7 Works [page 230]
Involved Roles [page 149]

7.1.4.1.4 Renewal of Keys for CMS/PKCS#7 Decryptor - Inbound

This use case covers all situations where private keys used by the tenant to decrypt messages from a sender (in our terminology: inbound messages) are changed. The renewal process ensures that the related public encryption key is changed at sender side that way that no downtime is required.

The CMS/PKCS7 decryptor uses a private key to decrypt a PKCS7/CMS encrypted payload. This private key is provided in the tenant keystore together with a X509 certificate. The decryptor uses an alias configured in the corresponding integration flow step to locate the private key in the keystore.

The following figure illustrates the communication path that is relevant for this use case.



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Sender is able to Encrypt Payload with the old Key and the new Key

1. Tenant administrator: Creates a new key pair.
2. Tenant administrator: Adds the new key pair and the corresponding certificate into the keystore. After this step has been performed, the decryptor accepts payloads encrypted by the old and the new public key.
3. Tenant administrator: Hands over the new certificate to the sender administrator.
4. Sender administrator: Exchanges the certificate.
5. Sender administrator: Informs the tenant administrator.
6. Tenant administrator: Removes the old key pair from the keystore after a time period (which at least corresponds to the guaranteed delivery time) so that he can be sure that no payload encrypted with the old pulic key has been sent.

Sender is only able to Encrypt Payload with one Key

This process is applicable as of version 1.6 of the Integration Designer.

1. Tenant administrator: Creates a new key pair/certificate.
2. Tenant administrator: Adds the new key pair and the corresponding certificate into the keystore. From now on, the decryptor accepts payloads encrypted by the old and new public key.
3. Tenant administrator: Hands over the new certificate to the sender administrator.
4. Sender administrator: Exchanges the certificate.

5. Sender administrator: Informs the tenant administrator about the preceding step.
6. Tenant administrator: Removes the old key pair/certificate from the payload security keystore after a time period (which at least corresponds to the guaranteed delivery time) so that he can be sure that no payload encrypted with the old pulic key is being sent.

Related Information

How PKCS#7 Works [page 230]
Involved Roles [page 149]

7.1.4.2 Security Artifact Renewal for OpenPGP

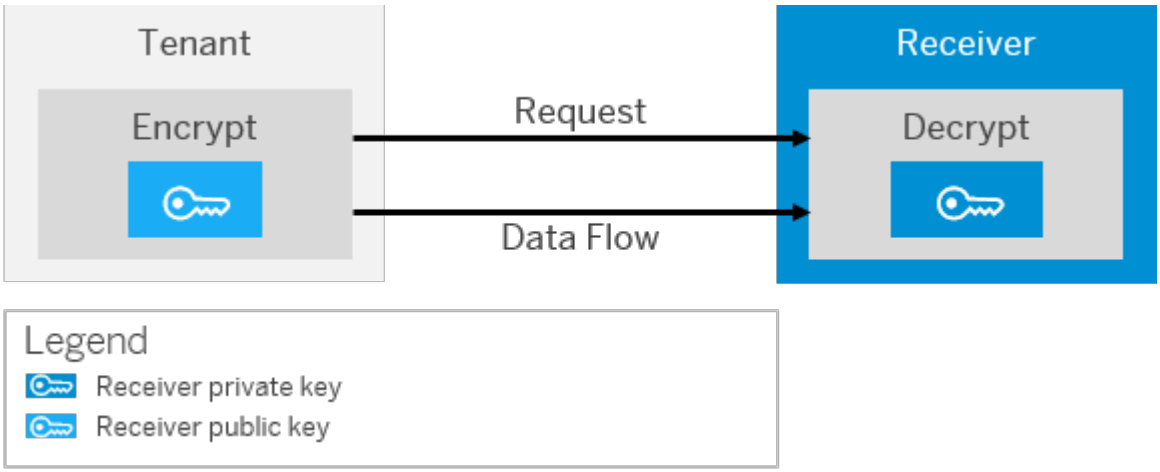
Related Information

How OpenPGP Works [page 234]
Renewal of OpenPGP Encryption Key - Outbound [page 175]
Renewal of OpenPGP Encryption Key - Inbound [page 178]
Renewal of OpenPGP Signer Key - Outbound [page 179]
Renewal of OpenPGP Signer Key - Inbound [page 181]

7.1.4.2.1 Renewal of OpenPGP Encryption Key - Outbound

In this use case, the tenant is the sender (outbound communication) and the receiver renews the encryption key.

The following figure illustrates the communication path that is relevant for this use case:



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

The renewal process depends on the capabilities of the customer receiver system.

Receiver Can Decrypt Payloads Encrypted by the Old Key and Payloads Encrypted by the New Key at One Point in Time

This use case requires a change of the integration flow if the user ID changes.

1. Receiver administrator: Creates a new PGP key pair.
2. Receiver administrator: Configures receiver system so that it can decrypt either payloads encrypted with the old key or payloads encrypted with the new key.
3. Receiver administrator: Provides the tenant administrator with the new PGP public key through a secure channel and informs the tenant administrator that for a certain period of time the receiver system will accept PGP messages encrypted either with the old key or with the new key.
4. Tenant administrator: Imports the new PGP public key into the PGP Public Keyring.
   If the public key has not been received through a secure channel, the tenant administrator checks that the new imported key has the correct fingerprint by comparing the fingerprint with a fingerprint provided via a trustworthy channel (phone, signed e-mail).
5. Tenant administrator: Informs the integration developer that he has to exchange the old encryption user ID with the new encryption user ID in the PGP encryptor step (only if the new key has a different user ID to the old key).
6. Integration developer adapts the integration flow (only if the new key has a different user ID to the old key).
   The integration developer adds the new user ID (or a part of the user ID) to the list of user IDs in the PGP encryptor step.
   The adapted integration flow has to be newly deployed. The old user ID must still be kept on the user ID list.
7. Tenant administrator: Deletes the old public key from the PGP Public Keyring and deploys the changed PGP Public Keyring.
   From now on the payloads are encrypted with the new public key.
8. After the agreed time period, the receiver administrator removes the old key pair and configures the receiver system so that from now on it can only receive payloads encrypted by the new key.

Receiver Can Only Decrypt Payloads Encrypted with the Same Key at One Point in Time and Accepts PGP Messages Containing Symmetric Keys Encrypted with Several Asymmetric Keys

This use case requires a change of the integration flow if the user ID changes.

1. Receiver administrator: Creates new PGP key pair.

2. Receiver administrator: Provides tenant administrator with the new PGP public key and informs the tenant administrator that the receiver system will only be able to decrypt PGP messages that are encrypted with the new key as of a certain date.
   Note that the PGP message may still contain an additional package containing the symmetric key encrypted with the old key.
3. Tenant administrator: Imports the new PGP public key into the PGP Public Keyring and checks that the new imported key has the correct fingerprint by comparing the fingerprint with a fingerprint provided via a trustworthy channel (phone, signed e-mail).
   This step is only necessary if the public key has not been received through a secure channel.
4. Tenant administrator: Informs the integration developer that he has to exchange the old encryption user ID with the new encryption user ID in the PGP encryptor step (only if the new key has a different user ID to the old key).
5. Integration developer adapts the integration flow (only if the new key has a different user ID to the old key).
   The integration developer adds the new user ID (or a part of the user ID) to the list of user IDs in the PGP encryptor step.
   The adapted integration flow has to be newly deployed. The old user ID must still be kept on the user ID list.
6. Tenant administrator: Deploys the changed PGP Public Keyring. From now on the payload is encrypted with the old key and the new key.
7. At the specified date the receiver administrator removes the old key from the receiver system.
8. After the specified date the tenant administrator removes the old public key from the PGP Public Keyring and deploys the changed PGP Public Keyring.
   From now on the payload is only encrypted with the new key.

Receiver Can Only Decrypt Payloads Encrypted with the Same Key at One Point in Time and Does Not Accept PGP Messages Containing Symmetric Keys Encrypted with Several Asymmetric Keys

This use case requires a downtime.

1. Receiver administrator: Creates new PGP key pair.
2. Receiver administrator and tenant administrator agree on a downtime.
3. Receiver administrator: Provides tenant administrator with the new PGP public key.
4. Tenant administrator: Imports the new public key into the PGP Public Keyring and checks that the fingerprint of the new PGP public key is correct by comparing the fingerprint with a fingerprint provided via a trustworthy channel (phone, signed e-mail). This step is only necessary if the public key was not received through a secure channel.
5. During downtime, the following happens:
   1. Tenant administrator: Informs the integration developer that he has to exchange the old encryption user ID with the new encryption user ID in the PGP encryptor step (only if the new key has a different user ID to the old key).
   2. Integration developer adapts the integration flow (only if the new key has a different user ID to the old key).
      The integration developer adds the new user ID (or a part of the user ID) to the list of user IDs in the PGP encryptor step.
      The adapted integration flow has to be newly deployed.
   3. Tenant administrator: Removes the old key from the PGP Public Keyring and deploys the PGP Public Keyring.

4. Receiver administrator: Exchanges the old key with the new key in the receiver system.

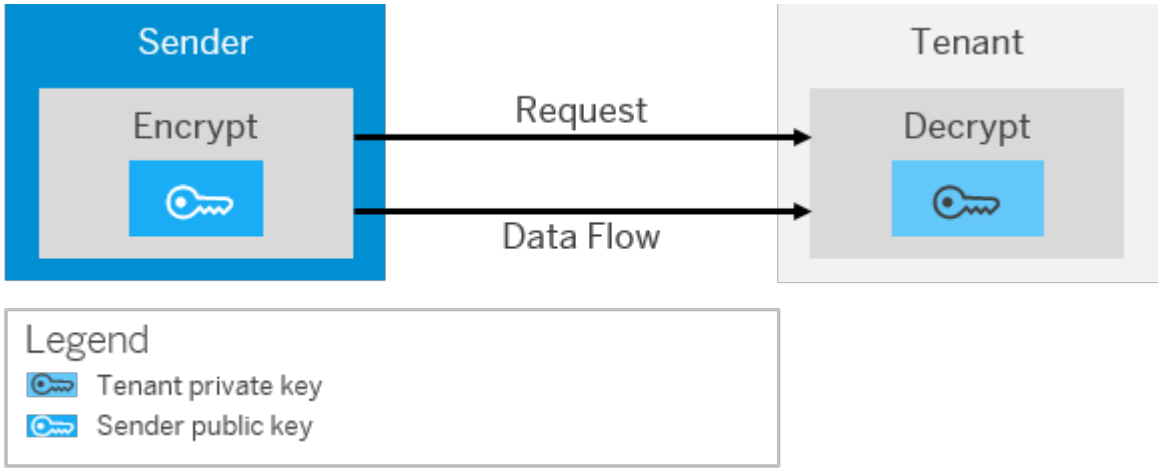
Related Information

Involved Roles [page 149]

7.1.4.2.2 Renewal of OpenPGP Encryption Key - Inbound

In this use case, the tenant is the receiver (inbound communication) and renews the encryption key.

The following figure illustrates the communication path that is relevant for this use case:



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

1. Tenant administrator: Creates a new PGP key pair with the same user ID as the old key in the PGP Secret Keyring.
2. Tenant administrator: Deploys the changed PGP Secret Keyring.
   From this moment on the PGP decryptor accepts payloads encrypted with the old key or payloads encrypted with the new public key.
3. Tenant administrator: Exports the new PGP public key.
4. Tenant administrator: Provides the sender administrator with the exported key via a secure channel and informs the sender administrator about the following:
   - For a certain period of time the tenant will be able to accept payloads encrypted with the old key or payloads encrypted with the new public key.
   - After this period only payloads encrypted with the new key are accepted.

5. Sender administrator: Exchanges the old key with the new key, so that from now on the sender system sends payloads encrypted with the new key.
6. After the specified period is over, the tenant administrator removes the old key pair from the PGP Secret Keyring and deploys the PGP Secret keyring.

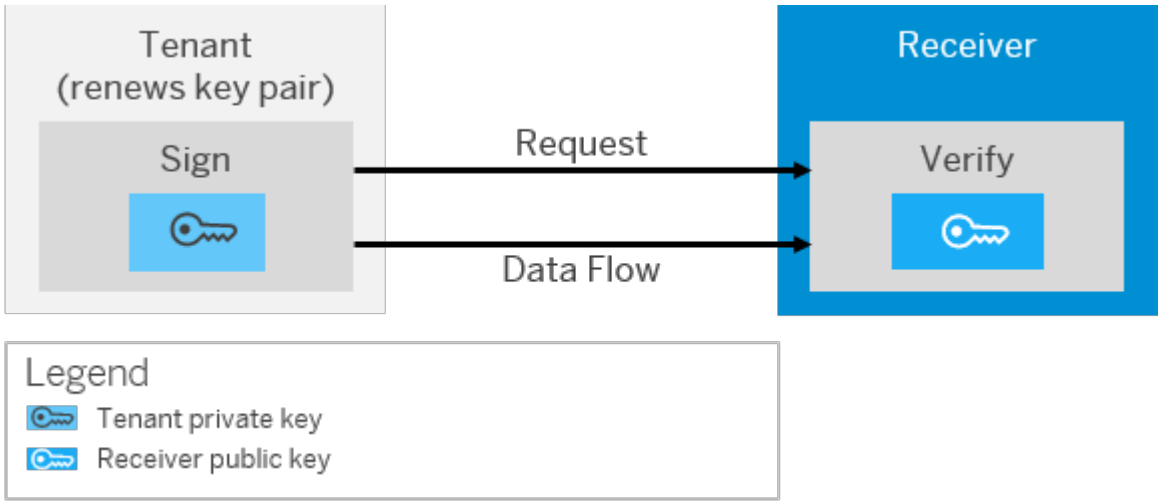
Related Information

Involved Roles [page 149]

7.1.4.2.3 Renewal of OpenPGP Signer Key - Outbound

In this use case, the tenant is the sender (outbound communication) and renews the signer key.

The following figure illustrates the communication path that is relevant for this use case:



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

The renewal process depends on the capabilities of the customer receiver system.

Receiver Can Verify Payloads Signed with the Old Key and Payloads Signed with the New Key at One Point in Time

1. Tenant administrator: Creates a new PGP key pair in the tenant's PGP Secret Keyring with the same user ID as the old key (without deploying the changed PGP Secret Keyring yet).
2. Tenant administrator: Exports the new PGP public key and provides receiver administrator with the new PGP public key via a secure channel. The tenant administrator informs the receiver administrator that as of a certain date the tenant will send payloads signed with the new key.
3. Receiver administrator: Configures the receiver system so that it can verify payloads signed with the old key or payloads signed with the new key.
4. On the specified date, the tenant administrator removes the old key from the PGP Secret Keyring and deploys the changed PGP Secret Keyring on the tenant.
   From now on the payloads are signed with the new key.
5. After the specified date, the receiver administrator removes the old key so that from now on the receiver system can only verify payloads signed by the new key.

Receiver Can Only Verify Payloads Signed with the Same Key at One Point in Time but Accepts PGP Messages with Two Signatures

1. Tenant administrator: Creates a new PGP key pair in the tenant's PGP Secret Keyring with the same user ID as the old key.
2. Tenant administrator: Deploys the changed PGP Secret Keyring on the tenant.
   From now on, the signed PGP message will contain two signatures, one from the old key and one from the new key.
3. Tenant administrator: Exports the new PGP public key and provides the receiver administrator with the new public key and informs the receiver administrator about the following:
   - For a certain period of time, PGP messages with two signatures will be sent.
   - After this period, PGP messages with one signature made by the new key will be sent
4. Before the specified period ends, the receiver administrator exchanges the old key with the new key and configures the receiver system so that it now verifies the signature with the new key.
5. After the specified period the tenant administrator removes the old key from the PGP Secret Keyring and deploys the changed PGP Secret Keyring on the tenant.
   From now on, PGP messages signed by the new key are sent.

Receiver Can Only Verify Payloads Signed with the Same Key at One Point in Time and Accepts Only PGP Messages with Exactly One Signature

This use case requires a downtime.

1. Tenant administrator: Creates a new PGP key pair in the tenant's PGP Secret Keyring with the same user ID as the old key.
2. Tenant administrator and receiver administrator agree on a downtime.

3. Tenant administrator: Exports the new public key from the PGP Secret Keyring and provides the receiver administrator with the exported public key.
4. During downtime, the following happens:
    1. Tenant administrator: Removes the old key from the PGP Secret Keyring and deploys the changed PGP Secret Keyring on the tenant.
    2. Receiver administrator: Exchanges the old key with the new key in the receiver system.

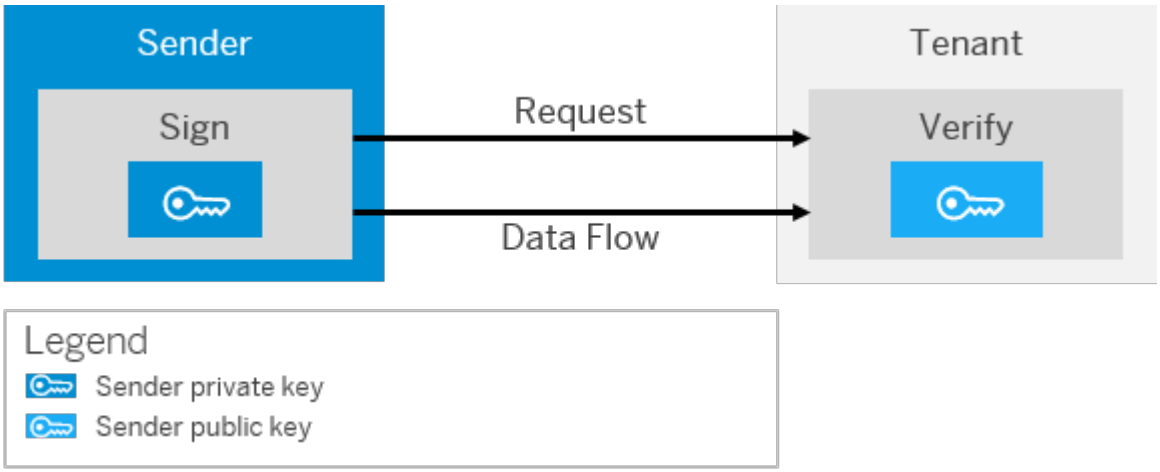
Related Information

Involved Roles [page 149]

7.1.4.2.4 Renewal of OpenPGP Signer Key - Inbound

In this use case, the tenant is the receiver (inbound communication) and the sender renews the signer key.

The following figure illustrates the communication path that is relevant for this use case:



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

1. Sender administrator: Creates a new PGP key pair for signing.
2. Sender administrator: Provides the tenant administrator with the new public key and informs the tenant administrator that as of a certain the payloads are signed with the new key.
3. Tenant administrator: Imports the public key into the PGP Public Keyring of the tenant and checks that the fingerprint of the new PGP public key is correct by comparing the fingerprint with a fingerprint provided via a trustworthy channel (phone, signed e-mail).
   This step is only necessary if the public key has not been received through a secure channel.

4. Tenant administrator: Informs the integration developer that he has to exchange the old encryption user ID with the new encryption user ID in the PGP decryptor step (only if the new key has a different user ID to the old key).
5. Integration developer adapts the integration flow (only if the new key has a different user ID to the old key).
   The integration developer adds the new user ID (or a part of the user ID) to the list of user IDs in the PGP decryptor step (so that the old and new user ID are contained in the list of signer user IDs).
   The adapted integration flow has to be newly deployed. The old user ID must still be kept on the user ID list.
6. Tenant administrator: Deploys changed PGP Public Keyring on the tenant.
   From now on, the tenant accepts payloads signed with the old key or with the new key.
7. On the specified date, the sender administrator configures the sender system so that it sends payloads signed with the new key from now on. The sender administrator removes the old key pair.
8. After the specified date, the tenant administrator removes the old public key from the PGP Public Keyring and deploys the changed PGP Public Keyring on the tenant.
   From now on the tenant only accepts payloads signed with the new key.

Related Information

Involved Roles [page 149]

7.1.4.3 Security Artifact Renewal for XML Digital Signature

7.1.4.3.1 Renewal of Keys for XML Digital Signature Signer - Outbound

This use case covers all situations where private keys (used by the tenant to sign outbound messages based on XML Digital Signature) are changed. The renewal process ensures that the related public verification key is changed at the receiver side that way that no downtime is required.

> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

The signer (when configured to use the XML Digital Signature standard) uses a private key to sign a payload. The private key is provided in the outbound keystore of the tenant. To locate the private key in the keystore, an alias is specified in the corresponding integration flow signer step.

The renewal process depends on whether the receiver system can verify signed data with different public keys at one point in time.

Receiver is able to Verify Payloads Signed by the old Key and Payloads Signed by the new Key at the Same Time

1. Tenant administrator: Creates a new key pair/certificate.
2. Tenant administrator: Provides the receiver administrator with the new certificate.
3. Receiver administrator: Configures the receiver system so that the receiver system can verify payloads signed by the old key or payloads signed by the new key.
4. Receiver administrator: Informs the tenant administrator that the receiver system can verify payloads signed by the old key or payloads signed by the new key.
5. Tenant administrator: Exchanges the old key pair/certificate with the new key pair/certificate in the keystore, keeping the old alias.
   From now on, the message is signed with the new key.
6. Tenant administrator: Informs the receiver administrator that certificate has been exchanged.
7. Receiver administrator: Removes the old certificate that way that from now on the receiver system can only verify payloads signed by the new key.

> **Note**
>
> This is the same process as to be applied for the CMS/PKCS#7 Signer.

Receiver is only able to Verify Payloads Signed by the Same Key at one Point in Time

This process implies a downtime.

1. Tenant administrator: Creates a new key pair/certificate.
2. Tenant administrator: Provides the receiver administrator with the new certificate.
3. Tenant administrator: Aggrees with the receiver administrator on a downtime.
   During certificate exchange no signed message is sent.
4. Tenant administrator: Informs the integration developer that the integration flow which signs the payload shall be undeployed.
5. Integration developer: Undeploys the integration flow (using the *Deployed Artifacts* editor tab of the Integration Operations feature).
   From now on, no further signed message is sent to the receiver system.
6. Integration developer informs tenant administrator about the preceding step.
7. Tenant administrator: Exchanges the old key pair/certificate with the new key pair/certificate, keeping the old alias in the keystore.
   From now on, the data are signed with the new key.
8. Tenant administrator: Informs the receiver administrator that no signed messages are being sent.
9. Receiver administrator: Exchanges the certificate in the receiver system.
10. Receiver administrator: Informs the tenant administrator that from now on the receiver system expects payloads signed by the new key.
11. Tenant administrator: Informs the integration developer that he can redeploy the integration flow.
12. Integration developer: Redeploys the integration flow (using the *Deployed Artifacts* editor tab of the Integration Operations feature).

Related Information

How XML Signature Works [page 232]
Involved Roles [page 149]

7.1.4.3.2 Renewal of Keys for XML Digital Signature Verifier - Inbound

This use case covers all situations where private keys used by a sender to sign messages sent to the tenant (in our terminology: inbound messages) based on XML Digital Signature are changed. The renewal process ensures that the related public verification key is changed at the tenant side that way that no or minimum downtime is required.

> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

The verifier (specified in the integration flow to use the XML Digital Signature standard) uses a public key to verifiy a payload signed by the sender. This public key has been imported into the tenant keystore as X509 certificate. The verifier uses an alias configured in the corresponding integration flow step to locate the public key in the keystore.

To apply the following process, the following prerequisites are met:

- The Integration Designer as of version 1.7 is used.
- The sender sends in the XML Signature information about the signer certificate (certificate, whole certificate chain, issuer DN and serial number of certificate, or combinations).
- The XML Signature Verifier step (in the integration flow) contains two aliases, one for the current certificate and one for the new certificate.
  If the alias for the new certificate does not yet exist in the XML Signature step then the integration developer has to add such an alias prior to the renewal process.

1. Sender administrator: Creates new key air/certificate.
2. Sender administrator: Provides the tenant administrator with the new certificate.
3. Tenant administrator: Adds the new certificate to the keystore taking care that for the alias the alias is used which already has been specified in the XML Signature Verifier step (of the related integration flow).
4. Tenant administrator: Informs the sender administrator that payloads signed with the new key can be sent.
5. Sender administrator: Configures sender system that way that from now on payloads signed with the new private key are being sent.
6. Sender administrator: Removes the old key pair/certificate.
7. Sender administrator: Informs the tenant administrator that the sender systems sends payloads signed with the new key.
8. Tenant administrator: Removes the old certificate from the payload security keystore after a certain time period (which at least corresponds to the guaranteed delivery time).

The time period is necessary to make sure that payloads signed with the old key are no longer in the SAP cloud system.

Related Information

How XML Signature Works [page 232]
Involved Roles [page 149]

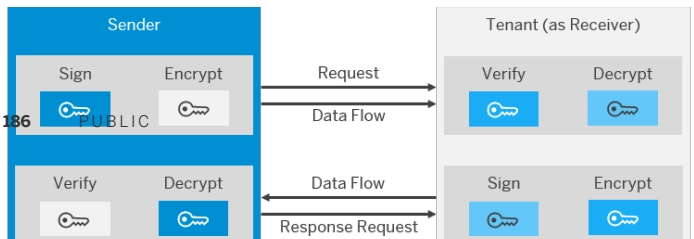
7.1.4.4 Security Artifact Renewal for WS-Security

Web services allow you to implement the request-response pattern. Web-Services Security (WS-Security) allows you to protect the request and response messages with digital signatures and encryption.

For Web services scenarios, there are separate security artifact renewal use cases for both request and response messages. WS-Security settings are configured in the sender and receiver SOAP adapters - depending on whether the tenant is a Web services provider or Web services consumer.

Inbound Communication (Tenant as WS Provider)

The following figure illustrates the communication paths that have to be considered when using WS-Security.


Sender
Sign
Encrypt
Verify
Decrypt
Request
Data Flow
Data Flow
Response Request
Tenant (as Receiver)
Verify
Decrypt
Sign
Encrypt

The corresponding settings on the tenant side are configured in the sender SOAP adapter.

The table below contains the security renewal use cases:

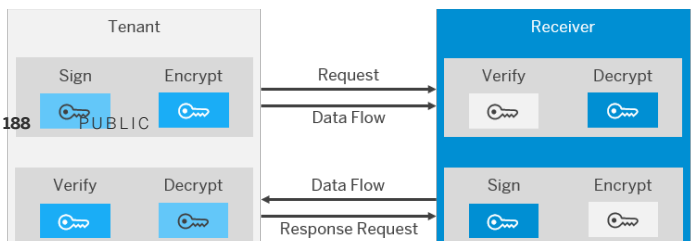
Security Renewal Use Cases (Inbound Communication)

Use Case	More information
Tenant verifies inbound request message.	Security Artifact Renewal for WS-Security (Tenant Verifies Inbound Request) [page 189]
Tenant decrypts inbound request message.	Security Artifact Renewal for WS-Security (Tenant Decrypts Inbound Request) [page 190]
Tenant signs inbound response message.	Security Artifact Renewal for WS-Security (Tenant Signs Inbound Response) [page 192]
Tenant encrypts inbound response message.	Security Artifact Renewal for WS-Security (Tenant Encrypts Inbound Response) [page 194]

> **Note**
>
> The terms *inbound* and *outbound* used in this section reflect the perspective of the tenant. *Tenant verifies inbound request message* refers to the request message sent from a sender system to the tenant (incoming message at tenant side).

Outbound Communication (Tenant as WS Consumer)

The following figure illustrates the communication paths that have to be considered when using WS-Security.


Tenant
Sign
Encrypt
Verify
Decrypt
Request
Data Flow
Data Flow
Response Request
Receiver
Verify
Decrypt
Sign
Encrypt

The corresponding settings at tenant side are configured in the receiver SOAP adapter.

The table contains the security renewal use cases:

Security Renewal Use Cases (Inbound Communication)

Use Case	More information
Tenant signs outbound request message.	Security Artifact Renewal for WS-Security (Tenant Signs Outbound Request) [page 195]
Tenant encrypts outbound request message.	Security Artifact Renewal for WS-Security (Tenant Encrypts Outbound Request) [page 196]
Tenant verifies outbound response message.	Security Artifact Renewal for WS-Security (Tenant Verifies Outbound Response) [page 198]
Tenant decrypts outbound response message.	Security Artifact Renewal for WS-Security (Tenant Decrypts Outbound Response) [page 200]

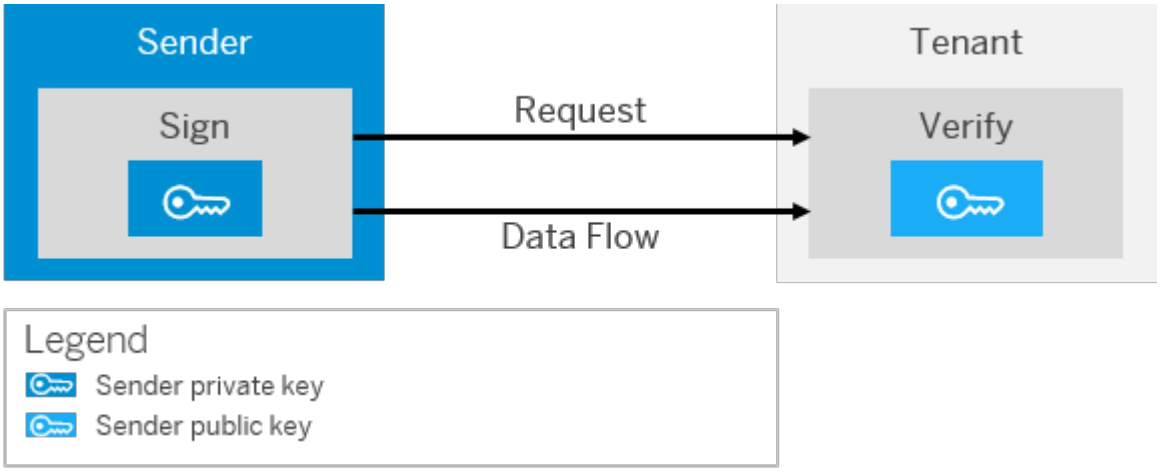
7.1.4.4.1 Security Artifact Renewal for WS-Security (Tenant Verifies Inbound Request)

This use case covers all situations where private keys used by a sender to sign messages sent to the tenant (in our terminology: inbound messages) are changed. The renewal process ensures that the related public verification key is changed on the tenant side so that no downtime is required.

> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

The verifier (specified in the integration flow as using the WS-Security standard) uses a public key to verify a request payload signed by the sender. This public key has been imported into the tenant keystore as an X.509 certificate. The verifier uses the information provided in the WS-Security payload to determine the public key in the keystore.

The following figure illustrates the communication path for this use case.



1. Sender administrator: Creates new key pair/certificate.
2. Sender administrator: Provides the tenant administrator with the new certificate.
3. Tenant administrator: Adds the new certificate to the tenant keystore.
4. Tenant administrator: Informs the sender administrator that payloads signed with the new key can be sent.
5. Sender administrator: Configures sender system to send payloads signed with the new private key from now on.
6. Sender administrator: Removes the old key pair/certificate.
7. Sender administrator: Informs the tenant administrator that the sender system sends payloads signed with the new key.
8. Tenant administrator: Removes the old certificate from the keystore after a certain time period (which must be at least the guaranteed delivery time).
   This time period is necessary to make sure that payloads signed with the old key are no longer in the system.

The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

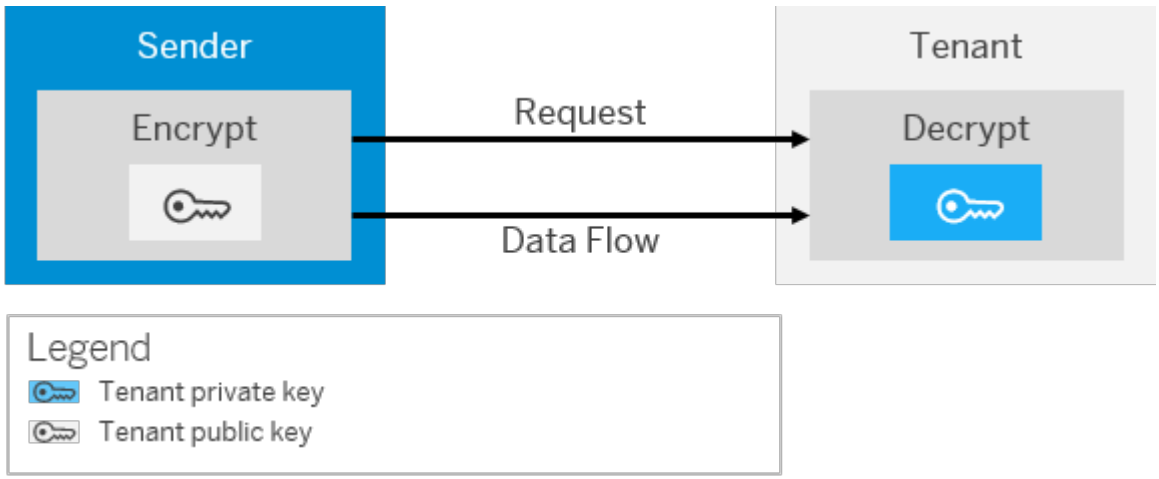
Related Information

How WS-Security Works [page 233]
Involved Roles [page 149]

7.1.4.4.2 Security Artifact Renewal for WS-Security (Tenant Decrypts Inbound Request)

This use case covers all situations where private keys used by the tenant to decrypt request messages from a sender (in our terminology: inbound messages) are changed. The renewal process ensures that the related public encryption key is changed on the sender side so that no downtime is required.

The following figure illustrates the communication path for this use case.



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

1. Tenant administrator: Creates new key pair/certificate.
2. Tenant administrator: Adds the new certificate to the keystore.
3. Tenant administrator: Provides the sender administrator with the new certificate and informs the sender administrator that payloads encrypted with the new key can be sent.
4. Sender administrator: Configures sender system to send payloads encrypted with the new public key/certificate from now on.
5. Sender administrator: Removes the old public key/certificate.
6. Sender administrator: Informs the tenant administrator that the sender system sends payloads encrypted with the new key.
7. Tenant administrator: Removes the old key pair/certificate from the keystore after a certain time period (which must be at least the guaranteed delivery time).
   This time period is necessary to make sure that payloads encrypted with the old key are no longer in the system.

The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

Related Information

How WS-Security Works [page 233]
Involved Roles [page 149]

7.1.4.4.3 Security Artifact Renewal for WS-Security (Tenant Signs Inbound Response)

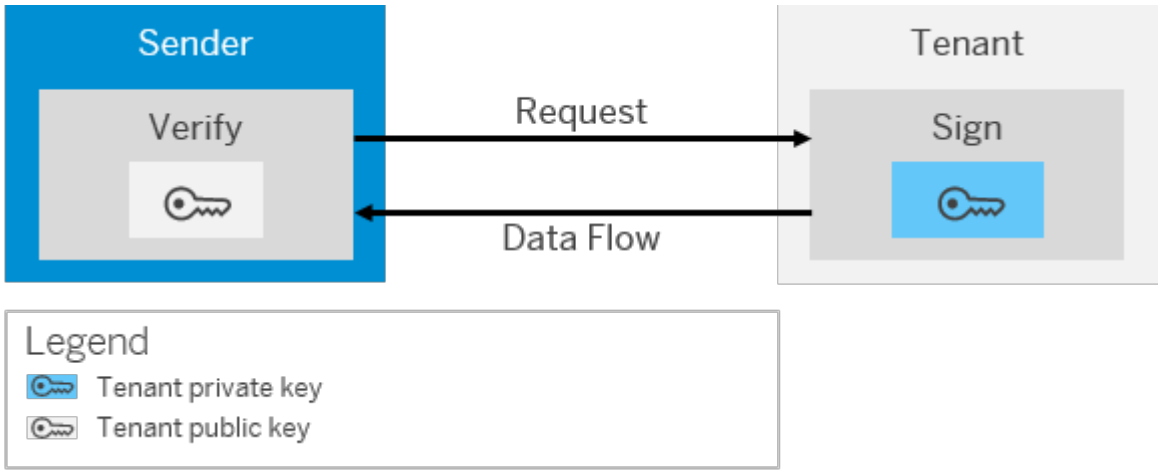
This use case covers all situations where private keys (used by the tenant to sign inbound response messages based on WS-Security) are changed. The renewal process ensures that the related public verification key is changed on the sender side (that receives the response message) so that no or only a minimum downtime is required.

The signer (if configured to use the WS-Security standard) uses a private key to sign a payload. The private key is provided in the tenant keystore. To locate the private key in the keystore, an alias is specified in the corresponding integration flow signer step.

The signed WS-Security data contains either the certificate of the public key corresponding to the private key or the issuer and serial version number of the certificate so that the receiver can easily determine the public key with which the signature must be verified.

The renewal process depends on whether the sender system (that sends the request message and receives the response message) can verify signed data with different public keys at the same time.

The following figure illustrates the communication path for this use case.



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Sender is Able to Verify Payloads Signed by the Old Key and Payloads Signed by the New Key at the Same Time

1. Tenant administrator: Creates a new key pair.
2. Tenant administrator: Provides the new certificate to the sender administrator.

3. Sender administrator: Configures the sender system so that it is able to verify payloads signed by the old key and payloads signed by the new key.
4. Sender administrator: Informs the tenant administrator that the sender system is able to verify payloads signed by the old key and payloads signed by the new key.
5. Tenant administrator: Exchanges the old key pair with the new key pair, keeping the old alias.
   From now on, outbound messages are signed with the new key.
6. Tenant administrator: Informs the sender administrator that the key pair has been exchanged.
7. Sender administrator: Removes the old key pair.
   From now on, the sender system can only verify payloads signed by the new key.

> **Note**
>
> The process is similar to the WS-Security Signer Outbound Request case, however, the role of the receiver administrator is replaced by the role of the sender administrator, and the receiver system is replaced by the sender system.

Sender is Only Able to Verify Payloads Signed by the Same Key at One Time

1. Tenant administrator: Creates a new key pair/certificate.
2. Tenant administrator: Provides the new certificate to the sender administrator.
3. Tenant administrator: Agrees a downtime with the sender administrator so that no signed messages are sent during the certificate exchange.
4. Sender administrator: Stops sending signed messages at the start of the agreed downtime.
5. Sender administrator: Exchanges the old certificate with the new certificate.
6. Sender administrator: Informs the tenant administrator that message sending has been stopped and that the certificate has been exchanged.
7. Tenant administrator: Ensures that there are no messages signed with the old key in the system.
8. Tenant administrator: Exchanges the old key pair/certificate with the new key pair/certificate, keeping the old alias in the keystore.
   From now on, data is signed with the new key.
9. Tenant administrator: Informs the sender administrator that the key pair/certificate has been exchanged.
10. Sender administrator: Starts sending messages.

The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

Related Information

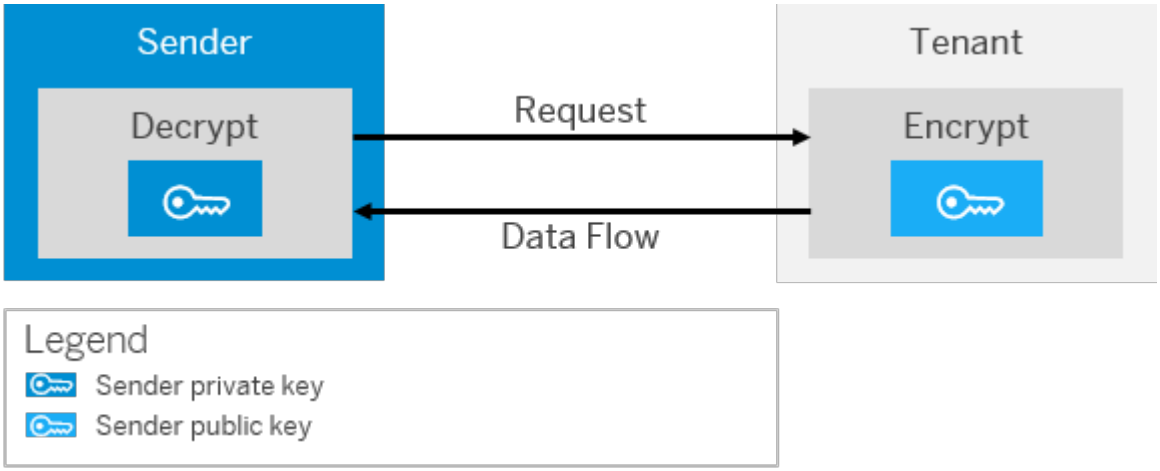
How WS-Security Works [page 233]
Involved Roles [page 149]

7.1.4.4.4 Security Artifact Renewal for WS-Security (Tenant Encrypts Inbound Response)

This use case covers all situations where a private key used to decrypt a response message (of an inbound request) is changed on the side of the sender of the response. The renewal process ensures that the related public encryption key is changed on the tenant side so that no or minimum downtime is required.

The WS-Security encryptor uses a public key to encrypt the payload of the inbound response message.

The following figure illustrates the communication path for this use case.



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Sender is Able to Decrypt Payloads Encrypted by the Old Key and Payloads Encrypted by the New Key at the Same Time

1. Sender administrator: Creates a new key pair/certificate.
2. Sender administrator: Configures the sender system so that it can decrypt payloads encrypted by the old key and payloads encrypted by the new key.
3. Sender administrator: Provides the tenant administrator with the new certificate and informs the tenant administrator that the sender system can verify payloads signed by the old key and payloads signed by the new key.
4. Tenant administrator: Exchanges the old key pair/certificate with the new key pair/certificate in the keystore, keeping the old alias.
   From now on, data is encrypted with the new key.
5. Tenant administrator: Informs the sender administrator that the certificate has been exchanged.

6. Sender administrator: Removes the old certificate so that from now on the sender system can only verify payloads signed by the new key.

Sender is Only Able to Decrypt Payloads Encrypted by the Same Key at One Time

1. Sender administrator: Creates a new key pair/certificate.
2. Tenant administrator: Agrees a downtime with the sender administrator so that no encrypted messages are sent during the certificate exchange.
3. Sender administrator: Stops sending messages at the start of the agreed downtime.
4. Sender administrator: Exchanges the old key pair/certificate with the new key pair/certificate.
5. Sender administrator: Provides the tenant administrator with the new certificate and informs the tenant administrator that message sending has been stopped and that the key pair/certificate has been exchanged.
6. Tenant administrator: Ensures that there are no messages encrypted with the old key in the system.
7. Tenant administrator: Exchanges the old certificate with the new certificate, keeping the old alias in the keystore.
   From now on, data is encrypted with the new key.
8. Tenant administrator: Informs the sender administrator that the key pair/certificate has been exchanged.
9. Sender administrator: Starts sending messages.

The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

Related Information

How WS-Security Works [page 233]
Involved Roles [page 149]

7.1.4.4.5 Security Artifact Renewal for WS-Security (Tenant Signs Outbound Request)

This use case covers all situations where private keys (used by the tenant to sign outbound messages based on WS-Security) are changed. The renewal process ensures that the related public verification key is changed on the receiver side so that no or only a minimum downtime is required.

The signer (if configured to use the WS-Security standard) uses a private key to sign a payload. The private key is provided in the outbound keystore of the tenant. To locate the private key in the keystore, an alias is specified in the corresponding integration flow signer step.

The signed WS-Security data contains either the certificate of the public key corresponding to the private key or the issuer and serial version number of the certificate so that the receiver can easily determine the public key with which the signature must be verified.

The renewal process depends on whether the receiver system can verify signed data with different public keys at the same time.

Receiver is Able to Verify Payloads Signed by the Old Key and Payloads Signed by the New Key at the Same Time

The same process applies as for the PKCS#7/CMS Signer in the same case.

Receiver is Only Able to Verify Payloads Signed by the Same Key at One Time

The same process applies as for the XML Digital Signer in the same case.

The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

Related Information

How WS-Security Works [page 233]
Renewal of Keys for CMS/PKCS#7 Signer - Outbound [page 165]
Renewal of Keys for XML Digital Signature Signer - Outbound [page 182]

7.1.4.4.6 Security Artifact Renewal for WS-Security (Tenant Encrypts Outbound Request)

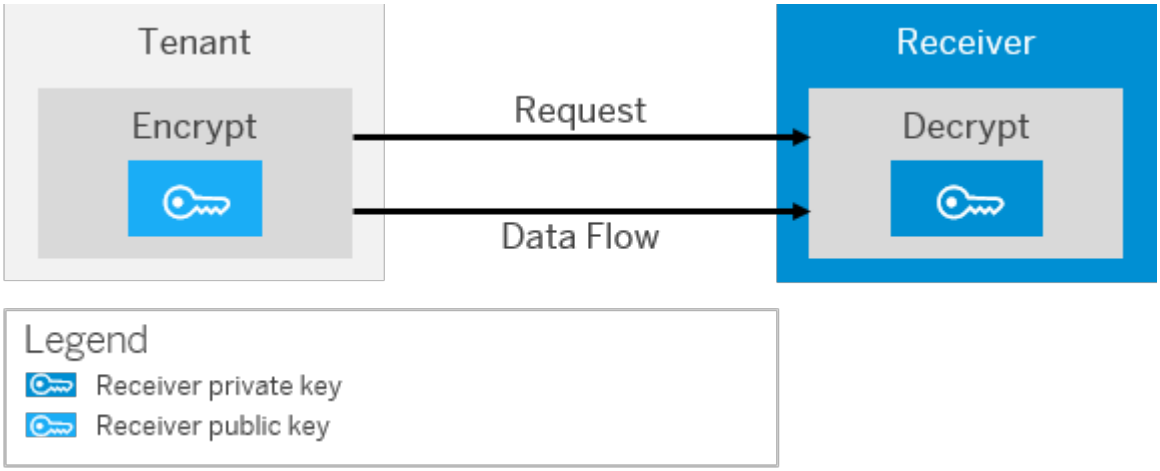
This use case covers all situations where a private key used for message decryption is changed by a receiver. The renewal process ensures that the related public encryption key is changed on the tenant side so that no or minimum downtime is required.

The encryptor uses a public key to encrypt a payload.

This public key is provided in the tenant keystore with an X.509 certificate. The encryptor uses an alias configured in the corresponding integration flow step to locate the public key in the keystore. The encrypted WS-Security data contains either the certificate of the public key corresponding to the private key or the issuer and serial version number of the certificate so that the receiver can easily determine the certificate and private key to be used for the decryption.

The renewal process depends on whether the receiver system can decrypt XML Encryption data with different public keys at the same time.

The following figure illustrates the communication path for this use case.



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Receiver is Able to Verify Payloads Encrypted by the Old Key and Payloads Encrypted by the New Key at the Same Time

1. Receiver administrator: Creates a new key pair/certificate.
2. Receiver administrator: Configures the receiver system so that it can decrypt payloads encrypted by the old key and payloads encrypted by the new key.
3. Receiver administrator: Provides the tenant administrator with the new certificate and informs the tenant administrator that the receiver system can verify payloads signed by the old key and payloads signed by the new key.
4. Tenant administrator: Exchanges the old key pair/certificate with the new key pair/certificate in the keystore, keeping the old alias.
   From now on, data is encrypted with the new key.
5. Tenant administrator: Informs the receiver administrator that the certificate has been exchanged.
6. Receiver administrator: Removes the old certificate so that from now on the receiver system can only verify payloads signed by the new key.

Receiver is Only Able to Verify Payloads Encrypted by the Same Key at One Time

1. Receiver administrator: Creates a new key pair/certificate.
2. Receiver administrator: Provides the tenant administrator with the new certificate.

3. Tenant administrator: Agrees a downtime with the receiver administrator so that no encrypted messages are sent during the certificate exchange.
4. Tenant administrator: Informs the integration developer that the integration flow that specifies the encryption of the payload must be undeployed.
5. Integration developer: Undeploys integration flow (using the *Deployed Artifacts* editor tab of the Integration Operations feature).
   From now on, no encrypted messages are sent to the receiver system.
6. Integration developer: Informs the tenant administrator about the preceding step.
7. Tenant administrator: Exchanges the old certificate with the new certificate, keeping the old alias in the keystore.
   From now on, data is encrypted with the new key.
8. Tenant administrator: Informs the receiver administrator that no encrypted messages are being sent.
9. Receiver administrator: Exchanges the key pair/certificate in the receiver system.
10. Receiver administrator: Informs the tenant administrator that from now on the receiver system expects payloads encrypted by the new key.
11. Tenant administrator: Informs the integration developer that the integration flow can be redeployed.
12. Integration developer: Redeploys the integration flow (using the *Deployed Artifacts* editor tab of the Integration Operations feature).

The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

Related Information

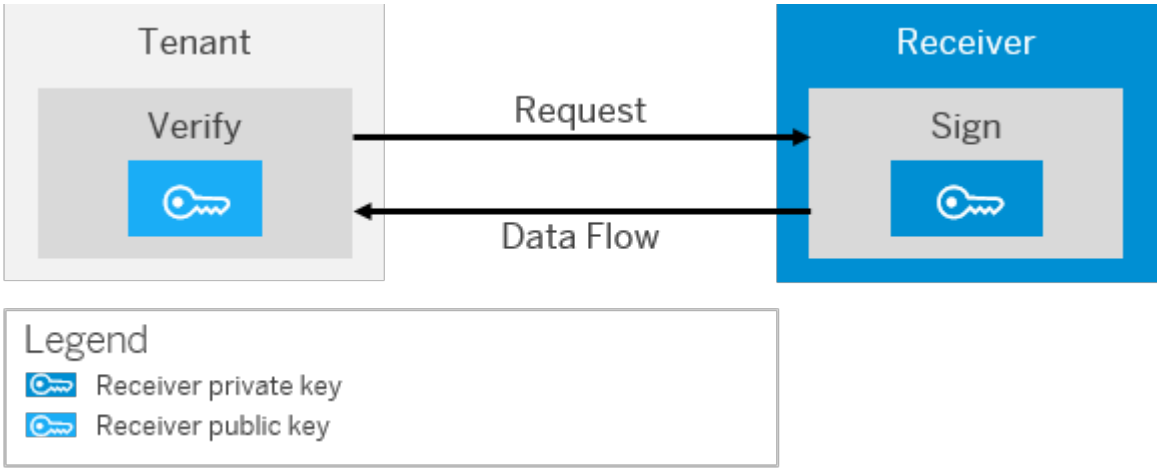
How WS-Security Works [page 233]
Involved Roles [page 149]

7.1.4.4.7 Security Artifact Renewal for WS-Security (Tenant Verifies Outbound Response)

This use case covers all situations where private keys used by a sender to sign response messages sent to the tenant are changed.

The WS-Security verifier uses a public key to verify a WS-Security response payload.

The following figure illustrates the communication path for this use case.



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

1. Receiver administrator: Creates a new key pair/certificate.
2. Receiver administrator: Provides the tenant administrator with the new certificate.
3. Tenant administrator: Adds the new certificate to the tenant keystore.
4. Tenant administrator: Informs the receiver administrator that payloads signed with the new key can be sent.
5. Receiver administrator: Configures receiver system to send payloads signed with the new private key from now on.
6. Receiver administrator: Removes the old key pair/certificate.
7. Receiver administrator: Informs the tenant administrator that the receiver system sends payloads signed with the new key.
8. Tenant administrator: Removes the old certificate from the keystore after a certain time period (which must be at least the guaranteed delivery time).
   This time period is necessary to make sure that payloads signed with the old key are no longer in the system.

The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

Related Information

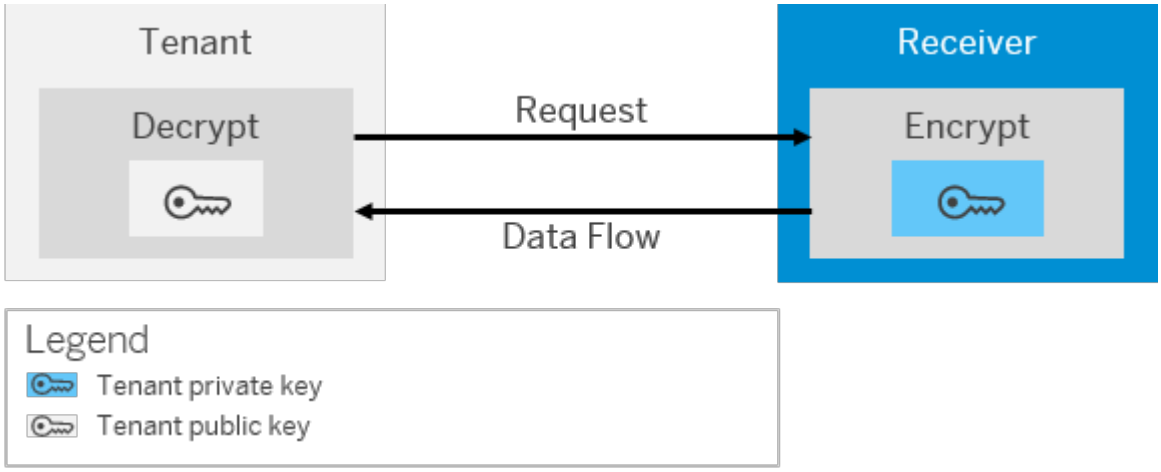
How WS-Security Works [page 233]
Involved Roles [page 149]

7.1.4.4.8 Security Artifact Renewal for WS-Security (Tenant Decrypts Outbound Response)

This use case covers all situations where private keys used by the tenant to decrypt a response message (from an outbound request) are changed. The renewal process ensures that the related public encryption key is changed on the side of the receiver of the request (that sends the response) so that no downtime is required.

The WS-Security decryptor uses a private key to verify a WS-Security response payload.

The following figure illustrates the communication path for this use case.



> **Note**
>
> It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

1. Tenant administrator: Creates a new key pair/certificate.
2. Tenant administrator: Adds the new certificate to the keystore.
3. Tenant administrator: Provides the receiver administrator with the new certificate and informs the receiver administrator that payloads encrypted with the new key can be sent.
4. Receiver administrator: Configures receiver system to send payloads encrypted with the new public key/certificate from now on.
5. Receiver administrator: Removes the old public key/certificate.
6. Receiver administrator: Informs the tenant administrator that the receiver system sends payloads encrypted with the new key.
7. Tenant administrator: Removes the old key pair/certificate from the keystore after a certain time period (which must be at least the guaranteed delivery time).
   This time period is necessary to make sure that payloads encrypted with the old key are no longer in the system.

The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

Related Information

How WS-Security Works [page 233]
Involved Roles [page 149]

7.2 Renewal of Keys Provided by SAP

To enable secure communication between the tenant and connected remote systems, the system keystore deployed on the tenant must contain up-to-date keys owned by the tenant administrator and SAP.

The certificate management features of the Web UI support the key renewal process described below.

> **Caution**
>
> You must exercise caution when renewing keys in order to avoid unplanned downtimes of your integration scenarios.
>
> The tenant administrator has to orchestrate the key renewal process together with the administrators of all involved sender and receiver systems. It may be necessary, for example, to agree on a specific downtime in certain cases.
>
> Note that it is the tenant administrator who should coordinate the whole process, as he or she knows all the administrators of the connected sender and receiver systems.
>
> Downtime can be avoided in certain cases. In order to schedule and set up a key renewal process correctly, see the topics linked below.
>
> We provide an example of a renewal process in a separate topic.

1. SAP prepares the new key pair 90 days before a key expires. The new key pair is imported into the customer tenant by SAP.
   The new key pair has the same alias as the key pair that is due to expire.
2. The tenant administrator can access the content of this keystore (Web UI Monitoring under *Manage Keystore* > *New SAP Keys*).
3. SAP informs the tenant administrator about the new key.
4. Shortly before the key expires, the tenant administrator downloads the X.509 certificate and certificate chain of the new key pair from the New SAP Keys keystore (Web UI Monitoring under *Manage Keystore* > *New SAP Keys*).
5. The tenant administrator informs the administrators of the connected sender and receiver systems that keys need to be renewed. If required, the administrators of the sender and receiver systems and the tenant administrator agree on a downtime during which the affected keys can be exchanged both in the sender and receiver systems and on the tenant.
6. The administrators of the sender and receiver systems update the keystores of their systems using the new certificate and certificate chain (provided to them by the tenant administrator).
   The tenant administrator activates the new key pair on the tenant with the certificate management features of the Web UI.

Related Information

Keys Provided by SAP [page 202]
Activating a New SAP Key Pair on the Tenant [page 202]
Example: Renewal of Key Pairs Provided by SAP (Avoiding Downtime) [page 203]
Renewal of SAP Keys Without Any Downtime [page 205]

7.2.1 Keys Provided by SAP

When SAP first provides a tenant to a customer, it delivers an X.509 certificate chain, which has to be renewed every 2 years.

SAP key pairs delivered with the tenant have the following aliases: sap_cloudintegrationcertificate, hcicertificate, or hcicertificate1.

You can use them in the following channels and integration flow steps:

- HTTPS outbound connections with client-certificate authentication
- Signature creator steps (XML Signature, CMS/PKCS#7, WS Security in SOAP adapter)
- Decryptor steps (CMS/PKCX#7, WS Security in SOAP adapter)

In all use cases, the corresponding X.509 certificate has to be made available to the sender or receiver system to enable the following steps:

- Configuring the required certificate-user mapping (HTTPS communication)
- Verifying the signature
- Encrypting the message

If the key pair is renewed without exchanging the certificate with the receiver or sender system, message processing will fail.

7.2.2 Activating a New SAP Key Pair on the Tenant

SAP Cloud Platform Integration comes with a set of features that facilitate the tenant administrator's task of renewing keys provided by SAP on the tenant.

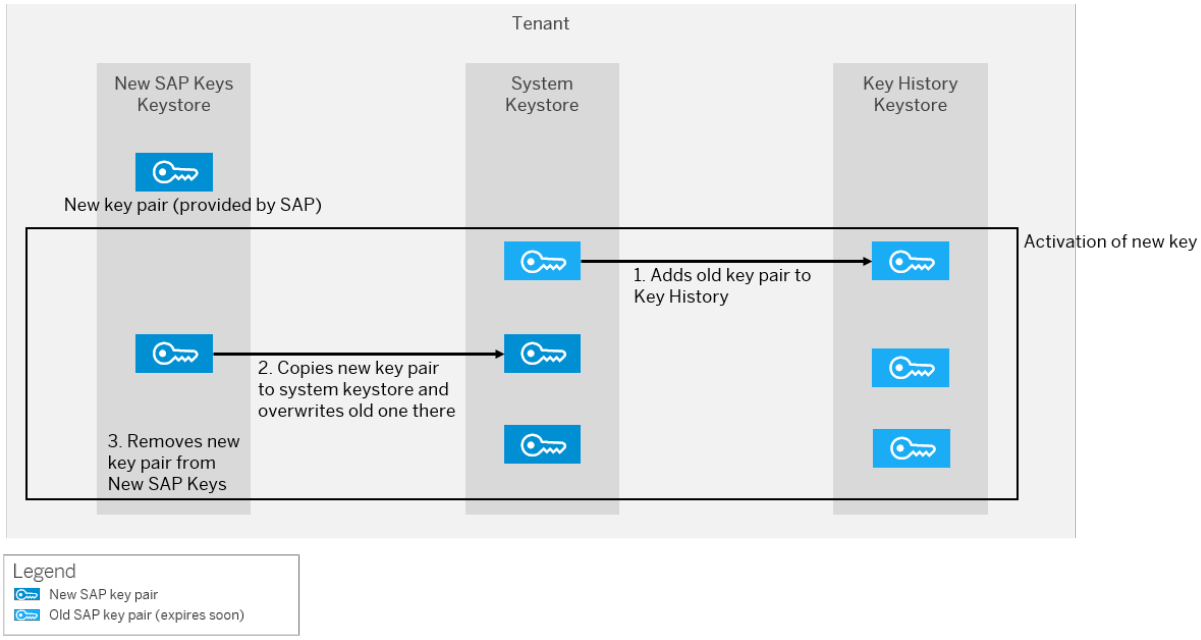
The following different keystore types enable the tenant administrator to renew keys in an efficient way:

- System keystore
  This keystore is located in the `system.jks` file provided with the tenant and contains all keys that are actively used by the deployed integration flows.
- New SAP Keys keystore
  This keystore contains keys already prepared by SAP before a key pair expires.
  As the key expiry date approaches, the tenant administrator can do the following:

- Download the new SAP keys from the KeyRenewal keystore to share them with the administrators of the connected back ends
- Activate the keys relevant for the tenant

- SAP Key History keystore
  This keystore contains old (expired) keys.
  If required, the tenant administrator can restore a key from the Key History keystore and use it to replace a key in the system keystore.

During the activation of an SAP key, the system performs three steps, as shown in the following figure.



1. The old key pair (which expires soon) is added by the system to the Key History keystore.
2. The old key pair (in the system.jks keystore) is overwritten by the new key pair from the New SAP Keys keystore.
3. In the New SAP Keys keystore, the new key pair is removed.

> **Note**
>
> To restore an old key pair, the tenant administrator can copy the key pair from the SAP Key History keystore to the New Keys keystore and activate it there.

7.2.3 Example: Renewal of Key Pairs Provided by SAP (Avoiding Downtime)

In the following example, an SAP key pair is used in the following steps and channels (in several integration flows):

- CMS signer
- CMS decryptor

- HTTPS outbound channel

In addition, the sender and receiver systems meet the requirement that they can cope with two certificates at the same time.

The tenant administrator refers to the general key renewal documentation and finds out that key renewal can be accomplished without any downtime if the following conditions apply:

- All sender systems decrypting the message are capable of sending CMS messages encrypted by several public keys.
- All receiver systems verifying the CMS signature are capable of verifying CMS signatures signed with several private keys.
- All receiver systems called by the outbound HTTPS channels can cope with different client certificates coming from the same tenant (by using multiple certificate-user mappings).

To prepare for the renewal process, the tenant administrator asks all the administrators of the sender and receiver systems whether their systems meet these requirements. If the answer is *yes*, the keys can be renewed by following three major steps:

1. The tenant administrator provides all administrators of the sender and receiver systems with the X.509 certificate and the X.509 certificate chain of the new key pair.
   The administrators of the sender and receiver systems install these certificates in their systems so that the following conditions are met:
   - The CMS decryptor decrypts the CMS message with the old and new public key (the symmetric key is actually encrypted twice with the old and new public key).
   - The CMS verifier can verify CMS signatures signed by the old or new public key.
   - The HTTPS server can cope with the old and new client certificate.
2. The tenant administrator overwrites the old SAP key pair (in the tenant's system keystore) with the new SAP key pair by activating the new SAP key pair.
3. The tenant administrator informs the administrators of the sender and receiver systems that the key has been renewed.
   The administrators then remove the old X.509 certificate from their system.

This process ensures that no downtime is necessary in message processing during key renewal.

Tip

You can see from this example that the tenant administrator has to have a good knowledge of the integration flow steps and channels where the key pair is used. Furthermore, the administrators of the sender and receiver systems have to know the capabilities of their systems.

If a sender/receiver administrator doesn't know whether the CMS decryptor component of his or her system is capable of creating a CMS Enveloped Data message where the symmetric key can be decrypted by two public keys, to be on the safe side we recommend that the sender and receiver system administrators and the tenant administrator agree on a downtime. During the downtime window, all involved parties adapt the configuration to the new key pair. Note that this downtime window can only be organized if the tenant administrator knows the administrators of the sender and receiver systems.

7.2.4 Renewal of SAP Keys Without Any Downtime

In certain cases it is possible to avoid a downtime in message processing when renewing keys. See the following use-case descriptions:

Renewal of Keys for CMS/PKCS#7 Signer - Outbound [page 165]

Renewal of Keys for CMS/PKCS#7 Decryptor - Inbound [page 173]

Renewal of OpenPGP Signer Key - Outbound [page 179]

Renewal of OpenPGP Encryption Key - Inbound [page 178]

Renewal of Keys for XML Digital Signature Signer - Outbound [page 182]

Security Artifact Renewal for WS-Security (Tenant Decrypts Inbound Request) [page 190]

Security Artifact Renewal for WS-Security (Tenant Signs Inbound Response) [page 192]

Security Artifact Renewal for WS-Security (Tenant Encrypts Outbound Request) [page 196]

Security Artifact Renewal for WS-Security (Tenant Decrypts Outbound Response) [page 200]

> **Note**
>
> These topics describe separate processes for renewing the various keys that are involved in an integration scenario using SAP Cloud Platform Integration. The topics cover situations where a key is owned either by the tenant administrator or by the administrator of the sender/receiver system connected to the tenant.
>
> To keep things simple, the individual topics describe idealized situations where a key pair is used in a **single** step or only in **one** communication channel. In real-life situations, however, a key pair is typically used in several integration flows, integration flow steps, and communication channels. The tenant administrator needs to know all the steps and channels where the key pair is used to be able to correctly define the key renewal process.

8 Support Tasks

When the customer-managed operating model is applied, the tenant cluster is managed by the customer. If support activites have to be performed on the tenant cluster by an SAP expert, the following steps are required.

As tenant administrator (at customer side) perform the following steps:

1. Ask ypur SAP contact to entitle an SAP expert as system development expert.
   SAP provides you with the corresponding user ID of the dedicated SAP expert.
2. **Temporaily** (for the time the support activities are going on) assign the authorization group *AuthGroup.SystemDeveloper* to this user.

Provide the tenant administrator (at customer side) with the user ID of the SAP expert and ask him or her to **temporaily** (for the time the support activities are going on) assign the authorization group *AuthGroup.SystemDeveloper* to this user.

Related Information

Overview of Authorization Groups [page 16]

9 Additional Features

9.1 Health checks and recommended actions for SAP Integration Advisor Node

Currently, the following health checks are covered for Integration Advisor node:

Integration Advisor State	The MBean checks the health of Integration Advisor. The threshold check is String 'Available'. In case Integration Advisor's health is good the check returns ' Available' and 'Unavailable' in case Integration Advisor's health is poor.
Availability Check	It checks the availability of the application via HTTP ping. The check is executed every second. Default values of 50 seconds for warning and 60 seconds for critical are currently assigned to this check. This means that if the ping response time is more than 50 or 60 seconds a warning or critical alert will be generated respectively.

9.2 Enabling SAP Solution Manager to Act as Additional Alert Consumer

You can enable SAP Solution Manager to display alerts.

Context

Currently, alerts are supported that are raised if no receiver can be determined during routing.

To set up this scenario, the SaaS administrator and the administrators of the involved SAP Solution Manager systems have to perform the following steps.

The following sequence focuses on the tasks relevant for the SaaS administrator.

> **Note**
>
> For more information on the tasks of the administrator of the SAP Solution Manager systems, see the documentation for SAP Solution Manager.
>
> Here's some additional information for using SAP Solution Manager with the integration platform:
>
> - Supported versions of SAP Solution Manager

Procedure

1. Provide the administrator of the SAP Solution Manager system (for example, at the customer side) with the tenant management node URL.

   The tenant management node URL is contained in the tenant mail that is sent out to the customer after the tenant provisioning process.
2. Create a user that enables the SAP Solution Manager system to connect to the integration platform as a client.
3. Assign the following role to this user: IntegrationOperationServer.consumealerts.
4. Provide the administrator of the SAP Solution Manager system with this user.
5. Register the SAP Solution Manager system as an alert consumer.
   a. Open the Integration Operations feature.
   b. Select the corresponding tenant, and on the *Tenant Configuration* tab enter the parameter `tmn.alertconsumers`.
   c. As a parameter value, add a comma-separated list of all SAP Solution Manager systems that should be registered as alert consumers.

      The SaaS administrator and the SAP Solution Manager administrator have to agree on the values that have to be entered as alert consumers, since the same values have to be entered in the configuration XML of the Solution Manager extractors.

Results

Once the connected SAP Solution Manager systems and tenants have been configured accordingly, alerts of the above-mentioned type are displayed in the Exception Management inbox of the involved SAP Solution Manager systems.

10 Concepts of Secure Communication

There are several options to protect the message exchange. You can secure the communication on transport level by selecting the HTTPS or SFTP protocol and installing specific authentication methods. In addition to that, you can set up methods to encrypt and decrypt the content of the message and to digitally sign and verify the message.

Related Information

Basics [page 209]
Security Elements [page 246]

10.1 Basics

Related Information

HTTPS-Based Communication [page 209]
SFTP-Based Communication [page 226]
Message-Level Security [page 227]
Certificate Management [page 240]

10.1.1 HTTPS-Based Communication

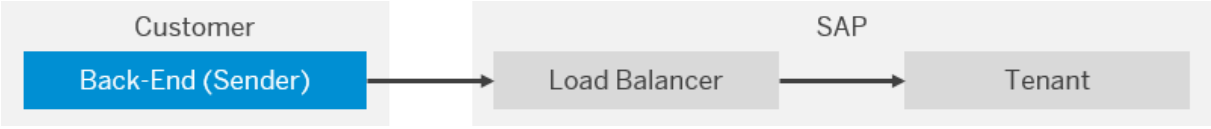
Related Information

Technical Landscape [page 210]
Authentication and Authorization Options (Inbound) [page 210]
Authentication Options (Outbound) [page 221]

Load Balancer Root Certificates Supported by SAP [page 225]

10.1.1.1 Technical Landscape

The following figures illustrate the general set up of components and communication paths.



Inbound Communication

Outbound Communication

For inbound SSL communication, a load balancer is interconnected between the sending customer back-end system and the tenant. Load balancer terminates inbound SSL connection (and starts a new one inside the VM landscape of SAP).

On the outbound side, a customer-specific tenant is connected to a customer back-end system. Therefore, all security settings for outbound message processing have to be configured tenant-specific.

The terms *inbound* and *outbound* reflect the perspective of SAP throughout this documentation.

- Inbound refers to message processing from a customer system to SAP (load balancer).
- Outbound refers to message processing from SAP (tenant) to a customer system.

Related Information

Virtual System Landscapes [page 7]

10.1.1.2 Authentication and Authorization Options (Inbound)

When a client calls a server using a secure communication channel, two different kinds of checks are performed subsequently.

- Authentication
  Verifies the identity of the calling entity.
- Authorization
  Checks what a user or other entity is authorized to do (for example, as defined by roles assigned to it). In other words, the authorization check evaluates the access rights of a user or other entity.

When a client calls a server, it is first authenticated and, in a subsequent step, the authorization check is performed.

We use **inbound** to refer to the communication direction when a sender system sends a message to the integration platform.

Combinations of Authentication and Authorization (Inbound)

For inbound communication based on HTTPS, the authentication and authorization options can be combined in a specific way.

Combination of Authentication/Authorization Options

Authentication Option ...	Can Be Used with the Following Authorization Option ...
Basic authentication The sender (client) authenticates itself against the server based on user credentials (user name and password). The HTTP header of the inbound message (from the sender) contains the user name and password.	**Role-based authorization** For this user, the authorizations are checked based on user-to-role assignments defined on the tenant.
Client-certificate authentication and certificate-to-user mapping The sender (client) authenticates itself against the server based on a digital client certificate. Furthermore, this certificate is mapped to a user (based on the information contained in a *Certificate-to-User Mapping* artifact deployed on the tenant). **Note** You can map multiple certificates to the same user (n:1 certificate-to-user mappings possible).	**Role-based authorization** For the user derived from the certificate-to-user mapping, the authorizations are checked based on user-to-role assignments defined on the tenant.
Client-certificate authentication (without certificate-to-user mapping) The sender (client) authenticates itself against the server based on a digital client certificate.	**Subject/Issuer DN authorization check of a certificate** In a subsequent authorization check, the permissions of the sender are checked on the tenant by evaluating the distinguished name (DN) of the client certificate of the sender.

Authentication Option ...	Can Be Used with the Following Authorization Option ...
OAuth Grants access to resources of SAP Cloud Platform Integration without the need to share passwords with the client. **Note** This option is supported for the following sender adapter types: SOAP (SOAP 1.x), SOAP (SAP RM), HTTPS.	**Role-based authorization**

More information: OAuth 2.0 Specification

Related Information

Protecting Applications with OAuth 2.0
Authentication Options (Inbound) [page 212]
Authorization Options (Inbound) [page 220]

10.1.1.2.1 Authentication Options (Inbound)

For inbound communication, different ways are supported how the sender can authenticate itself against Cloud Integration.

We use **inbound** to refer to the communication direction when a sender system sends a message to the integration platform.

- **Basic authentication**
  The calling entity is authenticated based on credentials (user name and password)
- **Client-certificate authentication and certificate-to-user mapping**
  The calling entity is authenticated based on a certificate, and the certificate is mapped to a user (for which the authorization check is executed in a subsequent step).
- **Client-certificate authentication** (without certificate-to-user mapping)
- OAuth 2.0
  OAuth allows you to set up authentication scenarios without the need to share credentials.
  More information on the concepts:
  Protecting Applications with OAuth 2.0
  OAuth 2.0 Specification

Related Information

Basic Authentication [page 213]
Client Certificate Authentication and Certificate-to-User Mapping (Inbound) [page 215]
Client Certificate Authentication (Inbound) [page 218]

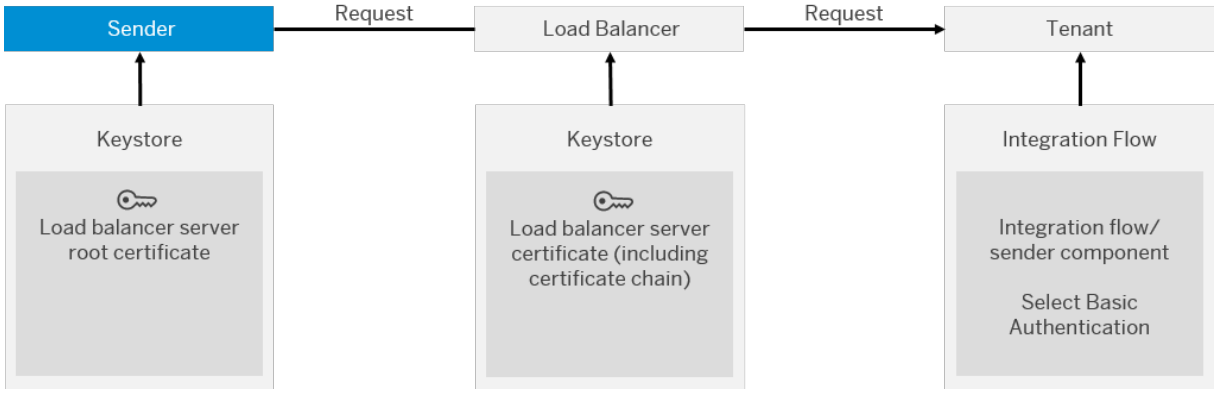
10.1.1.2.1.1 Basic Authentication

Basic authentication allows a client to authenticate itself against the server based on user credentials (user name and password).

> ⚠ Caution
>
> Consider that we do **not** recommend to use this option in productive scenarios because of the following security aspects:
>
> Basic authentication has the risk that authentication credentials, for example, passwords, are sent in clear text. Using TLS (transport-layer security, also referred to as Secure Sockets Layer) as transport-level encryption method (when using HTTPS as protocol) makes sure that this information is nevertheless encrypted on the transport path. However, the authentication credentials might become visible to SAP-internal administrators at points in the network where the TLS connection is terminated, for example, load balancers. If logging is not done properly at such devices, the authentication credentials might become part of log files. Also network monitoring tools used at such devices might expose the authentication information to administrators. Furthermore, the person to whom the authentication credentials belong (in the example above, the password owner) needs to maintain the password in a secure place.

How it Works

The following figure shows the setup of components required for inbound basic authentication.



These are the steps at runtime:

The HTTP header of the inbound message (from the sender) contains user name and password. To protect these credentials during the communication step, the connection is secured using TLS (SSL).

This includes a step where the load balancer authenticates itself as server against the sender based on a certificate. To enable this security measure, the keystore of the load balancer contains a server certificate signed by a certification authority. To be more precise, the keystore of the load balancer contains a complete certificate chain from (including all intermediate certificates). On the other side of the communication, the keystore of the connected sender system must contain the load balancer server root certificate. That is the certificate that identifies the certification authority (CA) that signed the load balancer's server certificate (on top of the certificate chain).

The other way round, the identity of the sender is checked by SAP evaluating the credentials (user and password) against the user.

It is also depicted in the figure that the authentication option needs to be activated for the corresponding integration flow.

Required Security Material

To enable the sender system to authenticate itself against the integration platform with basic authentication, a communication user has to be created for the sender.

The following figure provides an overview of the involved security artifacts and storage locations.

Certificates for Inbound Message Processing

Keystore	Certificate	Description
Sender keystore	Load balancer server root certificate (identifies CA that has signed the load balancer server certificate)	This certificate is required to identify the root CA at the top of the certificate chain that ultimately guarantees the trustability of the load balancer server certificate. In many cases, there is a multilevel setup of CAs so that a certificate is signed by an intermediate CA. The trustability of the intermediate CA is guaranteed by another intermediate CA one level higher, and so on, up to the root CA at the top of the **certificate chain**. In this case, it is necessary to assign the certificate chain to the certificate, to enable the connected component (which has imported only the root CA into its keystore) to evaluate the chain of trust.

Keystore	Certificate	Description
Load balancer keystore	Load balancer server certificate	This certificate is required to identify the load balancer as a trusted server (to which clients like the sender system can connect). This certificate is required for certificate-based authentication where the sender acts as the client. On the tenant side, this certificate is required to configure the authorization check.

For sakes of completeness, note that always a tenant keystore (not depicted in the figure) needs to be available to enable the system to do an additional outbound communication step that is required for technical purposes: The basic technical connectivity of a cluster is checked on a regular basis, as soon as the cluster is active. For this purpose, every 30 seconds the tenant management node sends an HTTPS request to an assigned runtime node via the load balancer. This simulates an external call to the runtime node. To enable this communication, a keystore needs to be deployed on the tenant, containing a valid client certificate that is accepted by the load balancer as well as the root certificate of the same. If this keystore is not available or contains an invalid certificate, the cluster will raise an error. The keystore and required certificate are provisioned by SAP together with the tenant.

10.1.1.2.1.2 Client Certificate Authentication and Certificate-to-User Mapping (Inbound)

This option includes an authentication step based on a digital client certificate and the mapping of the certificate to a user.

With a certificate-to-user mapping, a certificate is mapped to a user, and that way the user can be authenticated based on a certificate.

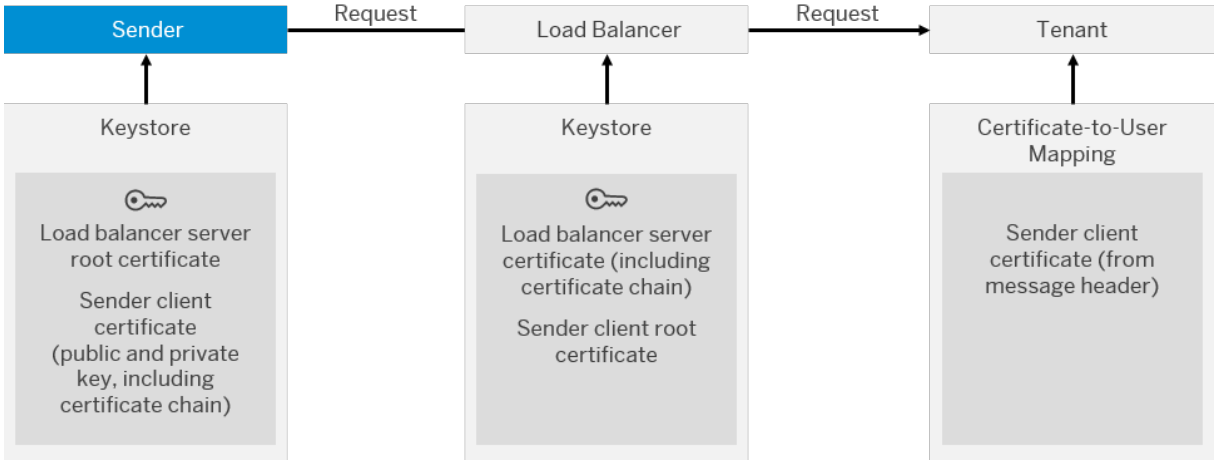
> **Note**
>
> Note that multiple certificates can be mapped to one user (n:1 certificate-to-user mappings possible).

Certificate-to-user mappings make sure that a user is always associated with the certificate as a whole, not only with one attribute of it (for example the common name (CN)). As different certificates can have the same CN, mapping only the CN to a user name bears the risk that different certificates can be mapped accidentally to the same user. Using certificate-to-user mappings circumvents this risk.

For the user defined that way, in a subsequent step, an authorization step is being executed.

How it Works

The following figure shows the complete setup of components and security artifacts required for this option.



When you have configured this authentication option, the authentication of the user is performed in the following way at runtime:

The TLS connection of the sender system and the integration platform is terminated and newly established by the load balancer. This means, that first the load balancer authenticates itself against (as server) the sender based on the load balancer server certificate. Vice versa, the sender authenticates itself against the load balancer as client using the sender client certificate.

To enable the sender to communicate that way with the load balancer, the sender administrator has to make sure that the sender client certificate is signed by one of the certification authorities that are supported by the load balancer.

The load balancer sets the following message header fields:

- `SSL_CLIENT_CERT`
  Contains the Base64-encoded sender client certificate.
- `SSL_CLIENT_USER`

When the authentication is been executed successfully, the load balancer writes the sender client certificate (base 64-encoded) into the message header (field SSL_CLIENT_CERT). The tenant then maps the sender client certificate to a user based on the certificate-to-user mapping which is deployed on the tenant.

> **Note**
>
> In a subsequent step, the authorization check for this user is then executed based on the user-to-role assignments (permissions) as configured in the Central Authorization Service (CAS).

Required Security Material

Certificates for Inbound Message Processing

Keystore	Certificate	Description
Sender keystore	Sender client certificate (public and private key; signed by CA with which the tenant has a trust relationship)	

Keystore	Certificate	Description
	Load balancer server root certificate (identifies CA that has signed the load balancer server certificate)	This certificate is required to identify the root CA at the top of the certificate chain that ultimately guarantees the trustability of the load balancer server certificate. In many cases, there is a multilevel setup of CAs so that a certificate is signed by an intermediate CA. The trustability of the intermediate CA is guaranteed by another intermediate CA one level higher, and so on, up to the root CA at the top of the **certificate chain**. In this case, it is necessary to assign the certificate chain to the certificate, to enable the connected component (which has imported only the root CA into its keystore) to evaluate the chain of trust.
Load balancer keystore	Load balancer server certificate	This certificate is required to identify the load balancer as a trusted server (to which clients like the sender system can connect). This certificate is required for certificate-based authentication where the sender acts as the client. On the tenant side, this certificate is required to configure the authorization check.
	Sender client root certificate	This certificate is required to identify the root CA at the top of the certificate chain that ultimately guarantees the trustability of the sender client certificate. There is a list of CAs that are supported by the load balancer.

For sakes of completeness, note that always a tenant keystore (not depicted in the figure) needs to be available to enable the system to do an additional outbound communication step that is required for technical purposes: The basic technical connectivity of a cluster is checked on a regular basis, as soon as the cluster is active. For this purpose, every 30 seconds the tenant management node sends an HTTPS request to an assigned runtime node via the load balancer. This simulates an external call to the runtime node. To enable this communication, a keystore needs to be deployed on the tenant, containing a valid client certificate that is accepted by the load balancer as well as the root certificate of the same. If this keystore is not available or contains an invalid certificate, the cluster will raise an error. The keystore and required certificate are provisioned by SAP together with the tenant.

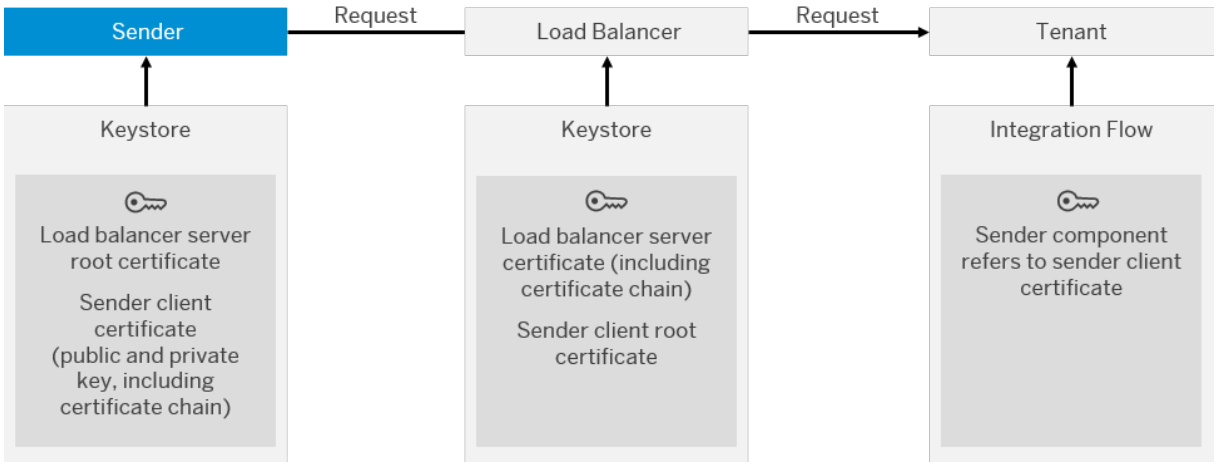
In addition to that, a certificate-to-user mapping has to be defined for the tenant.

10.1.1.2.1.3 Client Certificate Authentication (Inbound)

This option includes an authentication step based on a digital client certificate.

How it Works

The following figure shows the complete setup of components and security artifacts required for this option.



When you have configured this authentication option, the authentication of the user is performed in the following way at runtime:

The TLS connection of the sender system and the integration platform is terminated and newly established by the load balancer. This means, that first the load balancer authenticates itself against (as server) the sender based on the load balancer server certificate. Vice versa, the sender authenticates itself against the load balancer as client using the sender client certificate.

To enable the sender to communicate that way with the load balancer, the sender administrator has to make sure that the sender client certificate is signed by one of the certification authorities that are supported by the load balancer.

The load balancer sets the following message header fields:

- `SSL_CLIENT_CERT`
  Contains the Base64-encoded sender client certificate.
- `SSL_CLIENT_USER`

Required Security Material

Certificates for Inbound Message Processing

Keystore	Certificate	Description
Sender keystore	Load balancer server root certificate (identifies CA that has signed the load balancer server certificate)	This certificate is required to identify the root CA at the top of the certificate chain that ultimately guarantees the trustability of the load balancer server certificate. In many cases, there is a multilevel setup of CAs so that a certificate is signed by an intermediate CA. The trustability of the intermediate CA is guaranteed by another intermediate CA one level higher, and so on, up to the root CA at the top of the **certificate chain**. In this case, it is necessary to assign the certificate chain to the certificate, to enable the connected component (which has imported only the root CA into its keystore) to evaluate the chain of trust.
Load balancer keystore	Load balancer server certificate	This certificate is required to identify the load balancer as a trusted server (to which clients like the sender system can connect). This certificate is required for certificate-based authentication where the sender acts as the client. On the tenant side, this certificate is required to configure the authorization check.
	Sender client root certificate	This certificate is required to identify the root CA at the top of the certificate chain that ultimately guarantees the trustability of the sender client certificate. There is a list of CAs that are supported by the load balancer.

For sakes of completeness, note that always a tenant keystore (not depicted in the figure) needs to be available to enable the system to do an additional outbound communication step that is required for technical purposes: The basic technical connectivity of a cluster is checked on a regular basis, as soon as the cluster is active. For this purpose, every 30 seconds the tenant management node sends an HTTPS request to an assigned runtime node via the load balancer. This simulates an external call to the runtime node. To enable this communication, a keystore needs to be deployed on the tenant, containing a valid client certificate that is accepted by the load balancer as well as the root certificate of the same. If this keystore is not available or contains an invalid certificate, the cluster will raise an error. The keystore and required certificate are provisioned by SAP together with the tenant.

> **Note**
>
> In a subsequent authorization check, the permissions of the sender are checked on the tenant by evaluating the distinguished name (DN) of the client certificate of the sender. The client certificate of the sender is being passed through to the tenant by the load balancer (in the message header). To provide the tenant with the information on the correct client certificate to be expected from the sender, a corresponding setting has to be made in the related integration flow.

10.1.1.2.2 Authorization Options (Inbound)

For inbound HTTPS requests, two different ways to check the authorization of the caller can be configured.

We use **inbound** to refer to the communication direction when a sender system sends a message to the integration platform.

- **Role-based authorization**
  The permissions of the calling entity (user) are checked based on a user-to-role assignments configured in the associated identity provider.
  In the related sender adapter, you can assign the role based on which the inbound authorization is to be checked for the integration flow.
- **Subject/Issuer DN authorization check**
  The distinguished name (DN) of a certificate (associated with the calling entity) is checked.
  Subject/Issuer DN authorization check can be defined for individual integration flows.

Related Information

Role-Based Authorization [page 220]
Subject/Issuer DN authorization check [page 221]

10.1.1.2.2.1 Role-Based Authorization

This option allows you to define permissions for users in the connected identity provider (by default, SAP Identity Service) and to perform an authorization check based on these settings.

For HTTPS requests sent to Cloud Integration, it is checked if the role **ESBMessaging.send** is assigned to the user.

The permissions of the sending client are checked according to roles assigned to the user in the associated identity provider

User management (which includes the assignment of permissions to users) is performed by the tenant administrator using the SAP Cloud Platform Cockpit.

10.1.1.2.2.2 Subject/Issuer DN authorization check

It is checked (for a specific integration flow) if the subject/issuer distinguished name (DN) of the assigned certificate matches the incoming certificate.

If yes, this specific integration flow can be processed. The authorization check is performed based on the distinguished name (DN) of the client certificate. The DN has to be specified when configuring the relevant integration flow.

10.1.1.3 Authentication Options (Outbound)

For outbound communication through HTTPS (when the tenant sends a message to a receiver), the following authentication options are supported.

- **Basic authentication**
  The calling entity (tenant) is authenticated based on credentials (user name and password)
- **Client-certificate authentication**
  The calling entity (tenant) is authenticated based on a certificate.

Related Information

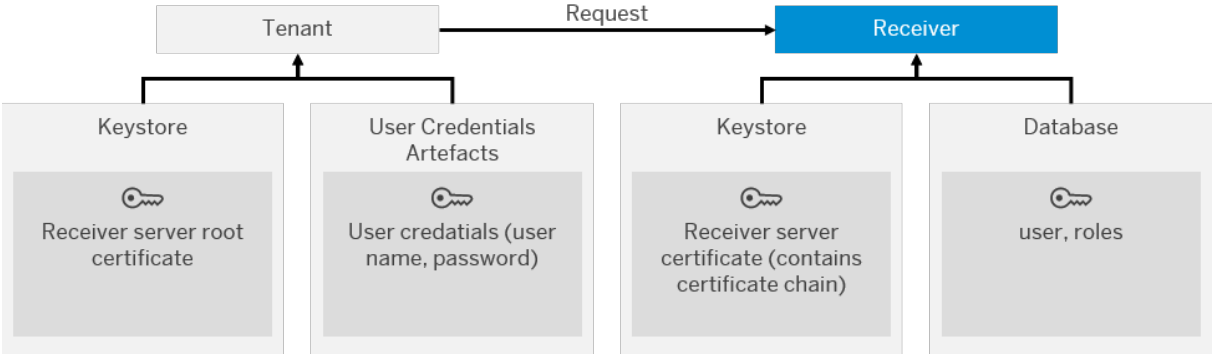
Basic Authentication [page 221]
Client Certificate Authentication (Outbound) [page 222]

10.1.1.3.1 Basic Authentication

Basic authentication allows a the tenant to authenticate itself against the receiver through credentials (user name and password).

How it Works

The following figure shows the setup of components required for this authentication option.



Basic authentication for HTTPS-based outbound calls works the following way:

1. The tenant (client) sends a message to the customer back-end system.
   The HTTP header of the message contains user credentials (name and password).
   To protect the user credentials during the communication step, the connection is secured using SSL.
2. The customer back-end authenticates itself as server against the tenant using a certificate (the customer back-end identifies itself as trusted server).
   To support this, the keystore of the customer back-end system must contain a server certificate signed by a certification authority. To be more precise, the keystore must contain the complete certificate chain. On the other side of the communication, the keystore of the connected tenant must contain the customer back-end server root certificate.
3. The tenant is authenticated by the customer back-end by evaluating the credentials against the user stored in a related data base connected to the customer back-end.

Required Security Material

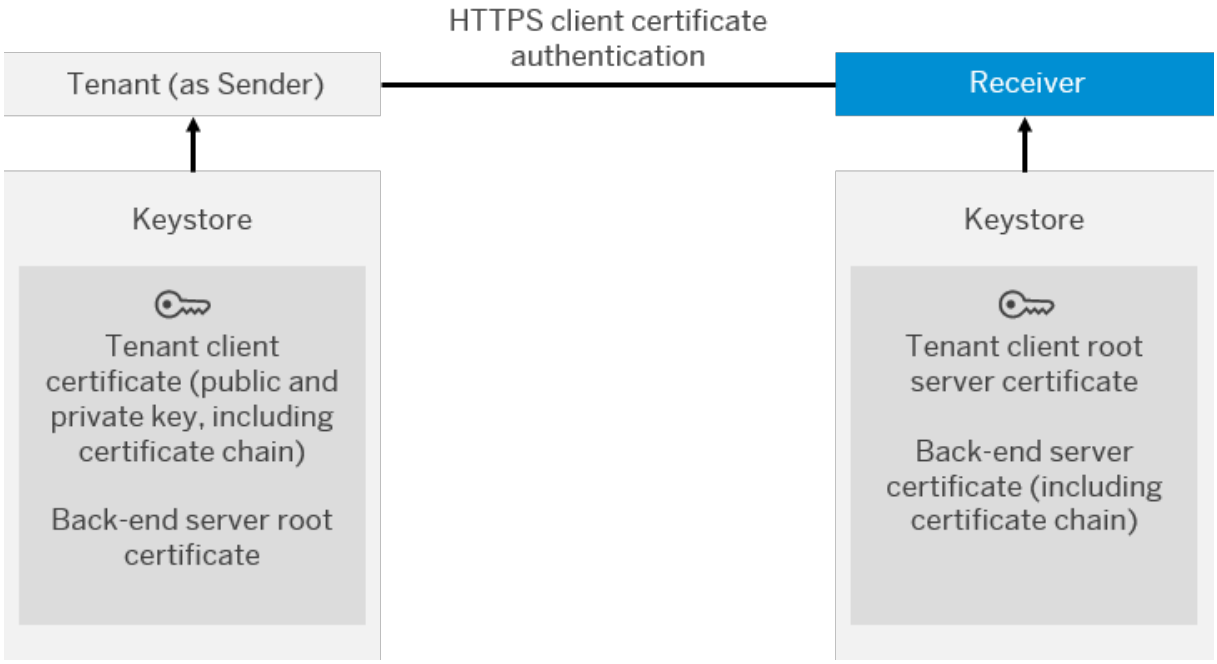
Certificates for Outbound Message Processing

Keystore	Security Element	Description
Keystore (tenant-specific)	Receiver server root certificate	This certificate is required to identify the root CA that is at the top of the certificate chain that ultimately guarantees the trustability of the receiver server certificate.
Receiver keystore	Receiver server certificate (signed by CA with which the tenant has a trust relationship)	This certificate is required to identify the receiver (to which the tenant connects as the client) as a trusted server.
User credentials artifact	User and password	With these credentials the tenant authenticates itself as client at the receiver system.

10.1.1.3.2 Client Certificate Authentication (Outbound)

The following figure shows the setup of components required for this authentication option.

How it Works



The tenant authenticates itself against the receiver based on a certificate.

This authentication option works the following way:

1. The tenant sends a message to the receiver.
2. The receiver authenticates itself (as trusted server) against the tenant when the connection is being set up. In this case, the receiver acts as server and the authentication is based on certificates.
3. Authentication of the tenant: The identity of the tenant is checked by the receiver by evaluating the client certificate chain of the tenant.
   As prerequisite for this authentication process, the client root certificate of the tenant has to be imported into the receiver keystore (prior to the connection set up).
   As CA who provides the root certificate, *Cyber trust Public Sure Server SV CA* is used.
   Steps 2 and 3 are referred to as mutual *SSL handshake*.
4. Authorization check: The permissions of the client (tenant) are checked in a subsequent step by the receiver.

Requried Security Artifacts

Certificates for Outbound Message Processing

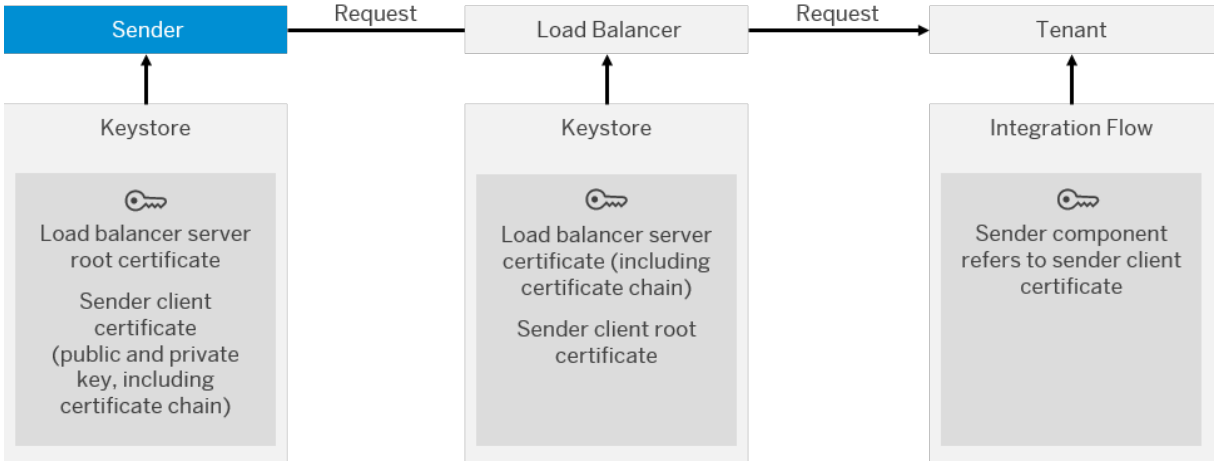
Keystore	Certificate	Description
Keystore (tenant-specific)	Tenant client certificate	This certificate is required to authenticate the tenant when calling the receiver system as the client. The certificate contains the public and private keys and has the certificate chain assigned to it. **Note** In many cases, there is a multilevel setup of CAs so that a certificate is signed by an intermediate CA. The trustability of the intermediate CA is guaranteed by another intermediate CA one level higher, and so on, up to the root CA at the top of the **certificate chain**. In this case, it is necessary to assign the certificate chain to the certificate, to enable the connected component (which has imported only the root CA into its keystore) to evaluate the chain of trust. The keystore is deployed on the tenant.
	Receiver server root certificate	This certificate is required to identify the root CA that is at the top of the certificate chain that ultimately guarantees the trustability of the receiver server certificate.
Receiver keystore	Receiver server certificate (signed by CA with which the tenant has a trust relationship)	This certificate is required to identify the receiver (to which the tenant connects as the client) as a trusted server.
	Tenant client root certificate (identifies CA that has signed the tenant client certificate)	This certificate is required to identify the root CA that is at the top of the certificate chain that ultimately guarantees the trustability of the tenant client certificate.

10.1.1.4 Load Balancer Root Certificates Supported by SAP

The load balancer supports a certain list of root certificates.

A system sending a message to the Cloud-based integration platform using HTTPS as secure transport channel is not directly connected to the tenant. Instead of this, a load balancer component is interconnected that terminates all inbound HTTPS requests, and re-establishes a new secure connection.

To set up a secure connection between a sender system and the integration platform, you therefore need to make sure that the sender system's keystore contains a client certificate that is signed by one of those certification authorities (CAs) that are trusted by the load balancer component of SAP.



The following list summarizes the **root certificates** that are supported by the load balancer.

> **Note**
>
> A specific certificate that identifies a certification authority (CA) is referred to as *root certificate*. Such a certificate is typically not signed by any other authority, as it is *at the root* of a certificate chain.

The load balancer component is owned by SAP, and you, the customer, don't need to care how it is configured. However, you need to make sure that the client certificate in your sender keystore is signed by one CA that is listed below.

> **Note**
>
> Please be aware that only **root certificates** are beeing imported into the *Keystore* of the *SAP Load Balancer* ! Therefore you as a customer must always assign the whole certificate chain to the certificate to enable the connected component to evaluate the chain of trust.

- AddTrust External CA Root
- Baltimore CyberTrust Root
- Certum CA
- DigiCert High Assurance EV Root CA
- DigiCert Global Root CA
- GeoTrust Global CA
- GeoTrust Primary Certification Authority - G3

- GlobalSign
- GlobalSign Root CA
- Go Daddy Class 2 Certification Authority
- Go Daddy Root Certificate Authority - G2
- Entrust Root Certification Authority
- Entrust.net Certification Authority (2048)
- Entrust Root Certification Authority - G2
- Equifax Secure Certificate Authority
- QuoVadis Root CA 2
- TC TrustCenter Class 2 CA II
- SwissSign Gold CA - G2
- SwissSign Platinum CA - G2
- SwissSign Silver CA - G2
- thawte Primary Root CA
- thawte Primary Root CA - G3
- VeriSign Class 1 Public Primary Certification Authority - G3
- VeriSign Class 3 Public Primary Certification Authority
- VeriSign Class 3 Public Primary Certification Authority - G5
- VeriSign Universal Root Certification Authority

10.1.2 SFTP-Based Communication

10.1.2.1 How SFTP Works

A tenant can connect **as SFTP client** to an SFTP server (the latter either hosted at SAP or in the customer landscape).

In a scenario using SFTP, an SFTP client connects to an SFTP server in order to perform one of the following tasks:

- SFTP client writes (*pushes*) a file to a file directory on an SFTP server.
- SFTP client reads (*pulls*) data from the SFTP server.
  Typically the SFTP client periodically reads files from the SFTP server.

Depending on the direction of data flow (whether the tenant reads data from the SFTP server or writes data to it), either an SFTP sender adapter or SFTP receiver adapter is involved.

Files are stored on the SFTP server in specific directories referred to as *mailboxes*. For each mailbox, a user is specified in order to control access to the data.

In certain cases, you have the option to choose between the following authentication options for SFTP connectivity in the SFTP (sender or receiver) adapter:

- User Name/Password

- Public Key

User Name/Password Authentication

The tenant connects to the server with a user and authenticates itself against the SFTP server with a password.

The user credentials (user name and password) are stored in a User Credentials artifact which has been deployed on the tenant prior to connection set up.

Public Key Authentication

In order to set up secure connection between the SFTP client and SFTP server, a combination of symmetric and asymmetric keys is applied.

- Symmetric (session) keys are used in order to encrypt and decrypt data within a data transfer session.
- Asymmetric key pairs (on client and server side) are used in order to encrypt and decrypt the session keys.

Symmetric and asymmetric keys are used by a client and a server exchanging data via SFTP in the following way:

1. The client connects to the server.
2. The server sends his public key to the client.
3. The client checks if the server is a trusted participant by evaluating a known_hosts file at client's side: if the server's public key is listed there-in, the identity of the server is confirmed.
4. The client generates a session key (to be used for one data transfer session).
5. The client encrypts the session key with the public key of the server.
6. The client sends the encrypted session key to the server. As public and private key of one party are mathematical correlated with each other, the server can decrypt the session key using its private key.
7. The session can now be continued in an encrypted way.
8. As part of the secure data transfer (using the session key exchanged by the step before), the client sends its public key to the server.
9. The server checks if the public key of the client is known to him (evaluating an authorized_keys file on the server side).
10. The server encrypts a random number with the client's public key and sends it to the client.
11. The client decrypts the random number with its private key and sends the unencrypted random number back to the server. That way, the client authenticates itself on server side.

10.1.3 Message-Level Security

Several standards are supported to protect the message content (message-level security).

Message-level security features allow you to digitally encrypt/decrypt or sign/verify a message (or both). The following standards and algorithms are supported.

Message-Level Security Options

Security Standard	Security Feature	Supported Algorithms
PKCS#7/CMS Enveloped Data and Signed Data PKCS#7/CMS provides a syntax for data that has cryptography applied to it, such as digital signatures or digital encryption. The CMS specification can be found at: http://tools.ietf.org/html/rfc5652 Digitally signing a message is based on the CMS type Signed Data. Digitally encrypting or decrypting the content of a message is based on the CMS type Enveloped Data.	Encryption/decryption of message content	Supported algorithms (by the symmetric key) for content encryption (format Cipher/Operation Mode/Padding Scheme): DESede/CBC/PKCS5Padding, DES/CBC/PKCS5Padding, AES/CBC/PKCS5Padding, ARCFOUR/ECB/NoPadding, Camellia/CBC/PKCS5Padding, RC2/CBC/PKCS5Padding, CAST5/CBC/PKCS5Padding.
	Signing/verification of payload	Supported algorithms for content signing (digest and encryption algorithm): SHA512/RSA, SHA384/RSA, SHA256/RSA, SHA224/RSA, SHA/RSA, RIPEMD128/RSA, RIPEMD160/RSA, RIPEMD256/RSA, MD5/RSA, MD2/RSA, RIPEMD160andMGF1/RSA-ISO9796-2-2-3, SHAandMGF1/RSA-ISO9796-2-2-3, SHA256withDSA, SHA224withDSA, SHA/DSA. The generated signature conforms to the CAdES-BES (CMS Advanced Electronic Signatures) signature standard according to the ETSI TS 101 733 V1.7.4 specification published at: http://www.etsi.org/deliver/etsi_ts/101700_101799/101733/01.07.04_60/ts_101733v010704p.pdf .
PKCS#7/CMS Enveloped Data and Signed Data	Encryption/decryption and signing/verification of payload	Supported algorithms (by the symmetric key) for content encryption (format Cipher/Operation Mode/Padding Scheme): DESede/CBC/PKCS5Padding, DES/CBC/PKCS5Padding, AES/CBC/PKCS5Padding, ARCFOUR/ECB/NoPadding, Camellia/CBC/PKCS5Padding, RC2/CBC/PKCS5Padding, CAST5/CBC/PKCS5Padding. Signature algorithms: SHA512/RSA, SHA384/RSA, SHA256/RSA, SHA224/RSA, SHA/RSA, RIPEMD128/RSA, RIPEMD160/RSA, RIPEMD256/RSA, MD5/RSA This is a subset of the algorithms that are supported for PKCS#7/CMS Enveloped Data and Signed Data. The generated signature **does not** conform to the CAdES-BES (CMS Advanced Electronic Signatures) signature standard.
Basic Digital Signature Option (*Simple Signer*)	Signing/verification payload	Supported algorithms for content signing (digest and encryption algorithm): SHA512/RSA, SHA384/RSA, SHA256/RSA, SHA224/RSA, SHA/RSA, RIPEMD128/RSA, RIPEMD160/RSA, RIPEMD256/RSA, MD5/RSA, MD2/RSA, RIPEMD160andMGF1/RSA-ISO9796-2-2-3, SHAandMGF1/RSA-ISO9796-2-2-3, SHA256withDSA, SHA224withDSA, SHA/DSA.

Security Standard	Security Feature	Supported Algorithms
Open Pretty Good Privacy (PGP)	Encryption/decryption of message content	Supported symmetric key algorithms for content encryption (symmetric key algorithms): CAST5 (128 bit key, as per [RFC2144]), Blowfish (128 bit key, 16 rounds), AES with 128, 192, and 256-bit key, Twofish with 256-bit key, DESede with 168-bit key DES is not supported.
	Encryption/decryption and signing/verification of the message	Supported signature algorithms for PGP signing: SHA-1, SHA224, SHA256, SHA384, SHA512, MD5, RIPE-MD/160
XML Signature	Signing/verification of payload	Supported signature algorithms: DSA/SHA1, RSA/SHA1, RSA/SHA256, RSA/SHA384, RSA/SHA512
XML Advanced Electronic Signature (XAdES) Supported XAdES forms: XAdES Basic Electronic Signature and XAdES Explicit Policy based Electronic Signature	Signing payload	The same signature algorithms as for XML Signature are supported.
WS-Security	Signing/verification of SOAP body Encryption/decryption of message content	The default signature algorithm is set by the data in the certificate, that is, one of the following: http://www.w3.org/2000/09/xmldsig#rsa-sha1 or http://www.w3.org/2000/09/xmldsig#dsa-sha1. The default signature digest algorithm is: http://www.w3.org/2000/09/xmldsig#sha1

Strong encryption is supported for the following algorithms:

- AES/CBC/PKCS5Padding
- Camellia/CBC/PKCS5Padding

For these algorithms, the key lengths 192 and 256 are possible.

Recommendations

Some algorithms (like MD2, MD5, DES or RC4) are still supported for legacy reasons, but they are not considered secure any more. We recommend that you check the official recommendations from National Institute of Standards and Technology (NIST) or European Union Agency for Network and Information Security (ENISA) for advice on algorithms and key strengths (for example, at: https://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-sizes-and-parameters-report).

Related Information

How PKCS#7 Works [page 230]
How XML Signature Works [page 232]
How WS-Security Works [page 233]
How OpenPGP Works [page 234]

10.1.3.1 How PKCS#7 Works

You have the option to digitally sign and encrypt message payloads based on PKCS#7/CMS Enveloped Data and Signed Data (PKCS stands for Public Key Cryptography Standards).

Signing and Verifying a Message

A digital signature ensures the authenticity of a message that way that it guarantees the identity of the signer and that the message was not altered after signing.

Digitally signing a message works the following way:

1. The sender calculates out of the message content a digest (or hash value) using a digest algorithm.
2. The sender encrypts the digest using a private key (type RSA or DSA). This is actually the signing step.

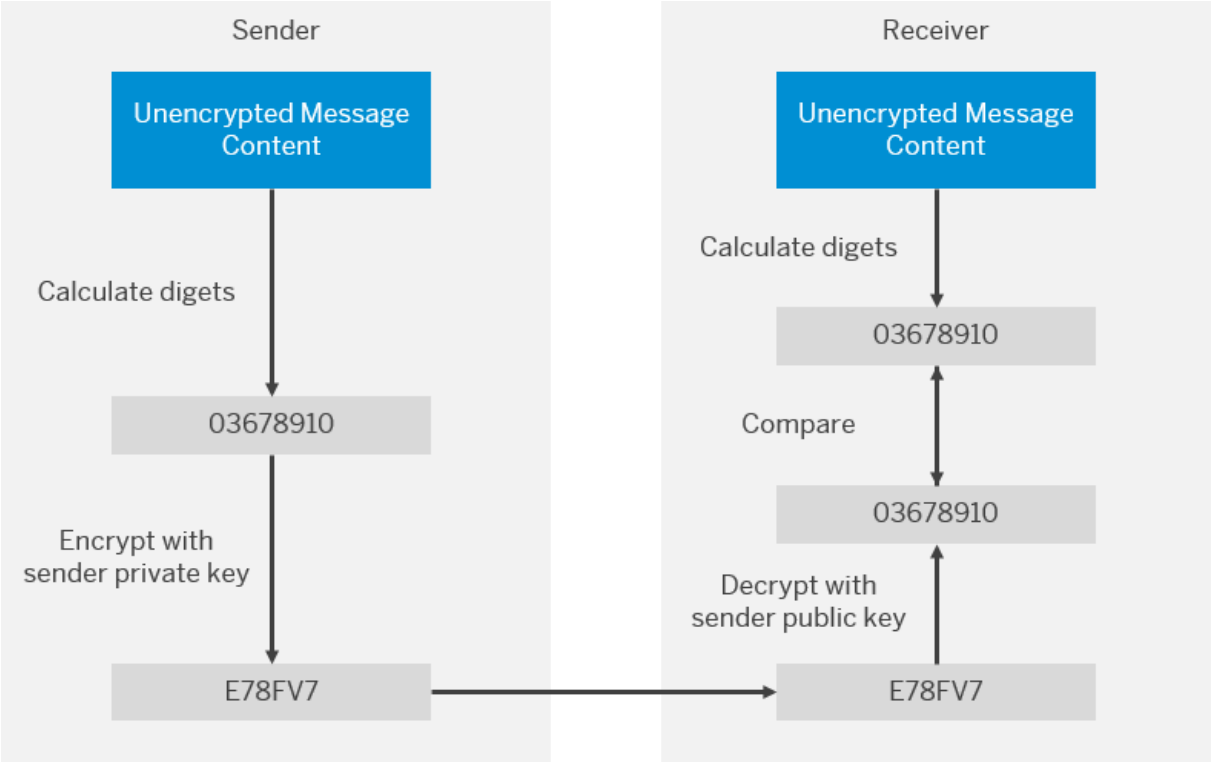
   > **Note**
   >
   > The following algorithms for content signing are supported (digest and encryption algorithm): SHA512/RSA, SHA384/RSA, SHA256/RSA, SHA224/RSA, SHA/RSA, RIPEMD128/RSA, RIPEMD160/RSA, MD5/RSA, MD2/RSA, RIPEMD160andMGF1/RSA-ISO9796-2-2-3, SHAandMGF1/RSA-ISO9796-2-2-3, SHA256withDSA, SHA224withDSA, SHA/DSA.

3. The sender sends the encrypted digest (which corresponds to the signature) together with the message content to the receiver.
4. The receiver decrypts the digest with the public key (which is related to the senders' private key). The public key has the type RSA or DSA.
5. The receiver calculates the digest out of the content of the message (which has been sent to it by the sender).
   The receiver uses the same digest algorithm which the sender had used.

   > **Note**
   >
   > PKCS#7 ensures that the digest algorithm is transferred together with the signature of the message and therefore available for the receiver.

   This calculation is based on the message content. In case the message content has been transferred encrypted, a decryption step is needed before this step.
6. The receiver compares the decrypted digest (from the sender) with the one calculated at receiver side. In case both values (digests) are identical, the signature is verified.

The following figure illustrates the process of digitally signing and verifying a message.



Encrypting and Decrypting the Content of a Message

Digital encryption allows you to encode the content of a message in such a way that only authorized parties can read it.

Digital encryption works two-stage based on symmetric and asymmetric key technology:

1. The sender encrypts the content of the message using a symmetric key.

   > **Note**
   >
   > The following algorithms for content encryption (by the symmetric key) are supported (format Cipher/Operation Mode/Padding Scheme): DESede/CBC/PKCS5Padding, DES/CBC/PKCS5Padding, AES/CBC/PKCS5Padding, ARCFOUR/ECB/NoPadding, Camellia/CBC/PKCS5Padding, RC2/CBC/PKCS5Padding, CAST5/CBC/PKCS5Padding.

2. The sender encrypts the symmetric key using a public key.

   > **Note**
   >
   > To encrypt the symmetric key, a public key of type RSA (with the cipher – or algorithm – RSA/ECB/PKCS1Padding) is used for each recipient.

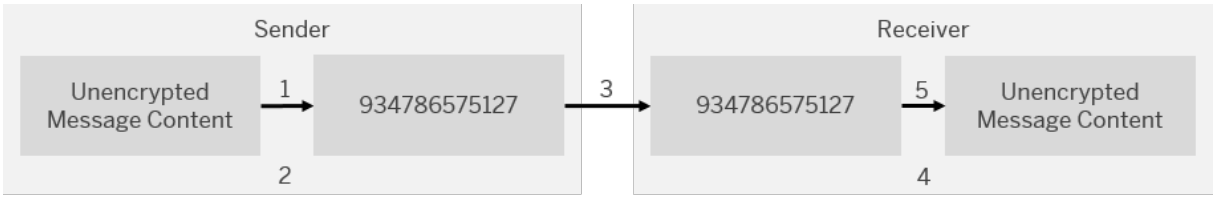
3. The sender sends the encrypted message and the encrypted symmetric key to the receiver.
4. The receiver decrypts the symmetric key using a private key (which is related to the public key used by the sender).

> **Note**
>
> For this decryption step a private key of type RSA is needed.

5. The receiver decrypts the content of the message using the decrypted symmetric key.

> **Note**
>
> Strong encryption is supported for the following algorithms:
>
> - AES/CBC/PKCS5Padding
> - Camellia/CBC/PKCS5Padding
>
> For these algorithms also the key lengths 192 and 256 are possible.

The following figure illustrates the process of digitally encrypting and decrypting the content of a message.



10.1.3.2 How XML Signature Works

A digital signature ensures the authenticity of a message that way that it guarantees the identity of the signer and that the message was not altered after signing. You have the option to digitally sign and validate a message based on the XML Signature standard (issued by the W3C consortium). Applying this standard means that the digital signature of a document itself is stored as an XML element.

XML Signature can be applied to any XML document.

The following options for XML Signature are supported:

Options to Apply XML Signature

Option	Description
Enveloped Signature	Digital signature/validation is applied to XML element that contains the signature as an element (the Signature element). Using this option, the digital signature is part of the XML document to be signed/validated.
Enveloping Signature	Digital signature/validation is applied to content within an Object element which is part of the Signature element. That way, the Signature elements acts as an envelope for the signed content. Using this option, the digital signature is part of the XML document to be signed/validated.

> **Note**
>
> You configure the usage of XML Signature in the related integration flow.

When applying XML Signature the following signature algoriths are supported: DSA/SHA1, RSA/SHA1, RSA/SHA256, RSA/SHA384, RSA/SHA512

When applying XML Signature the following canonicalization methods are supported: C14N, C14NwithComments, exc-C14N, exc-C14NwithComments

Background Information

In a simplified view, when configured correctly, digitally signing a message based on XML Signature implies the following main steps:

1. The sender of the message canonicalizes the XML message content that is to be signed.
   Canonicalization transforms the XML document to a standardized (reference) format. This step is required because an XML document can have more than one valid representations. Calculating a digest out of two different representations of the same document (according to step 2) results in different digests (or hash values). This would make the whole signing/validating process invalid.
2. Out of the canonicalized XML document, a digest is calculated using a digest algorithm.
3. The sender builds up a *SignedInfo* element that contains the digest.
4. The sender canonicalizes the *SignedInfo* element.
5. The sender builds a second digest for the *SignedInfo* element which contains the first digest.
6. The sender encrypts the digest using its private key.
7. The sender builds up the *SignatureValue* element which contains the encrypted digest from step 5 (the signature).
8. The message is sent to the receiver.

Digitally verifying (validating) a message based on XML Signature works the following way:

1. The receiver decrypts the encrypted digest (which is part of the *SignatureValue* element of the received message) using the sender's public key.
2. The receiver calculates the digest out of the *SignedInfo* element of the message.
3. The receiver compares the two digests that result out of steps 1 and 2.
   That way it is the authenticity of the sender is checked.
4. The receiver canonicalizes the XML message content.
5. The receiver calculates the digest out of the XML message content.
6. The receiver compares the digest that results from the canonicalized message content with that one contained in the *SignedInfo* element of the message.
   That way, it is made sure that the content of the message has not been altered during message processing.

10.1.3.3 How WS-Security Works

Messages can be protected according to the WS-Security standard.

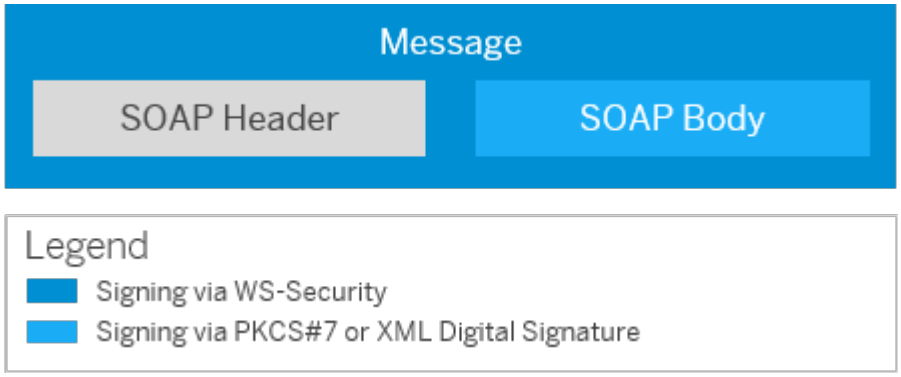
There are the following options:

- Digitally sign a message (and the other way round to verify a signed message)
- Digitally sign a message and to encrypt the message content (and the other way round to verify a message and to decrypt the message content)

Siging a Message

Signing a message (SOAP body) based on the WS-Security is an additional feature with regard to signing/verifying on payload level based on the following standards: PKCS#7, XML DigitalSignature (see figure below).

> **Note**
>
> For more information on the WS-Security standard see https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss .



10.1.3.4 How OpenPGP Works

You can use Open Pretty Good Privacy (Open PGP) to digitally sign and encrypt messages.

OpenPGP gives you the following options to protect communication at message level:

- You can encrypt a payload.
- You can sign and encrypt a payload.

OpenPGP does not support signing without encryption or just verifying without decryption. The tenant expects either an encrypted payload or a signed and encrypted payload.

During runtime, the encryptor/signer processor signs and encrypts the body of the inbound message and returns the resulting OpenPGP message in the body of the outbound message.

The required keys are stored in OpenPGP keyrings. The following types of keyrings exist:

PGP Keyrings

Type of Keyring	Description
PGP secret keyring	Contains the public/private key pairs of the sender. It can contain multiple key pairs, each identified by a user ID. The private key is protected with a passphrase. For PGP secret keyrings deployed on tenants, the same passphrase has to be used to access all private keys of the PGP secret keyring.
PGP public keyring	Contains the public keys (related to the private keys that are stored in the PGP secret keyring of the communication partner).

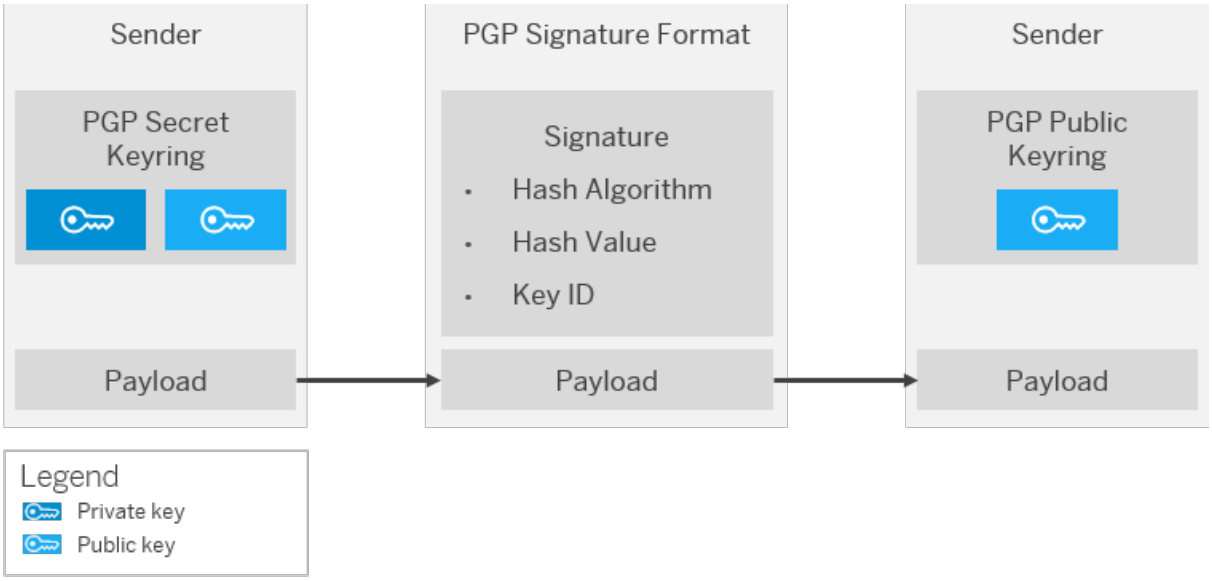
OpenPGP Signing/Verifying

A digital signature ensures the authenticity of a message by guaranteeing the identity of the signer and that the message has not been altered since signing.

A message is digitally signed and verified as follows:

1. The sender calculates a digest (or hash value) from the message content using a digest algorithm.
   The following hash algorithms are supported for PGP signing:
   - For DSA key: SHA-1, SHA224, SHA256, SHA384, SHA512
   - For RSA key: MD5, SHA-1, RIPE-MD/160, SH256, SHA384, SHA512, SHA224
2. The sender encrypts the digest using a private key (type RSA or DSA). This is the actual signing step.
   The private key is looked up in the sender's PGP secret keyring.
3. The encrypted hash value, together with the hash algorithm that has been used, is written to the signature element that is sent to the receiver together with the payload (as PGP signature format). The key ID of the signer of the private key is also written to the PGP signature format.
4. The receiver obtains the PGP signature format.
5. The receiver selects the key ID from the signature and uses the key ID to look up the right public key in the receiver's PGP public keyring. This is the public key that corresponds to the private key used to sign the payload.
   In addition, the receiver checks whether the user ID (associated with the key ID) corresponds to an allowed user.
6. The receiver decrypts the hash value (and verifies the payload) using the public key.

The following figure illustrates the concept.



OpenPGP Encrypting/Decrypting

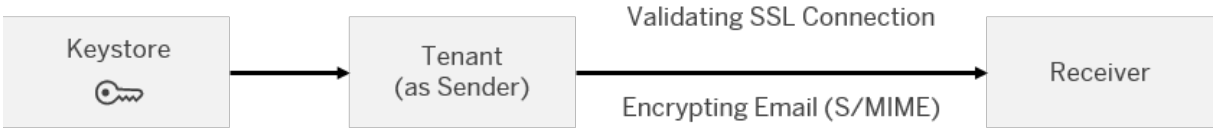
Digital encryption allows you to encode the content of a message in such a way that only authorized parties can read it.

A message is digitally encrypted and decrypted as follows:

1. The sender generates a symmetric key.
2. The sender encrypts the payload with the symmetric key.
   The following symmetric key algorithms for content encryption (symmetric key algorithms) are supported: TripleDES (168bit key derived from 192), CAST5 (128 bit key, as per [RFC2144]), Blowfish (128 bit key, 16 rounds), AES with 128, 192, and 256-bit key, Twofish with 256-bit key
   DES is not supported.
3. The sender looks up a public PGP key in the PGP public keyring.
4. The sender encrypts the symmetric key using the public PGP key (from the PGP public keyring).
   You can use the following key types to encrypt the symmetric key: RSA and Elgamal (DAS is not supported).
5. The sender writes the encrypted symmetric key and the key ID into the Encryption Info element of the message.
   The key ID is used to identify the public key used for encryption (as the PGP public keyring can contain more than one public key).
   The Encryption Info element is sent to the receiver, together with the encrypted payload.
6. The receiver obtains the message and, based on the key ID (in the Encryption Info element), looks up the correct private key (associated with the public key used to encrypt the payload) in the PGP secret keyring.
   A passphrase is required to access the private key.
7. The receiver decrypts the symmetric key with the private key from the PGP secret keyring.
8. The receiver decrypts the payload with the symmetric key.

There is an option to compress data before the encryption step. The following compression algorithms are supported: ZIP [RFC1951], ZLIB [RFC1950], BZip2.

The following figure illustrates the concept.



The runtime supports the following features:

- Signing with several private keys (the resulting OpenPGP message then contains several signatures).
- Encryption with several public keys.
  More precisely, the symmetric encryption key can be encrypted by several public keys (the resulting OpenPGP message then contains several *Public Key Encrypted Session Key* packets).
- Optional: OpenPGP compression and base 64 output or input.
- OpenPGP allows you to apply two different kinds of keys: primary keys and subkeys. (For simplicity, these are not differentiated in the figures above.)
  When you generate OpenPGP keys, a primary key with at least one subkey is created. Only the primary key can be used for certification, that is, to certify the trustworthiness of other keys. In addition, the primary key is also typically used to sign payloads. The subkey is used to encrypt payloads.

OpenPGP Message Format Specification

The OpenPGP message format is specified at http://tools.ietf.org/html/rfc4880 . An OpenPGP message is composed of a sequence of packets. The following table contains the most important packet types.

OpenPGP Message Format - Packets

Packet Type	Description
Public Key Encrypted Session Key	Session key encrypted with a public key, key ID of the public key, and public-key algorithm
Signature	Binding between a public key and some data. There are several types of signature packets: The certification, direct key, and subkey binding signature can be self-signed. The version 4 signature packet may also contain meta-information about the signature such as creation time, issuer, or key expiration time. The version 3 signature is deprecated.
Symmetric Key Encrypted Session Key	A symmetric key (also called session key) encrypted with a symmetric key; a symmetric algorithm is used. This packet is not supported.

Packet Type	Description
One-Pass Signature	Placed at the beginning of the message before the data. It contains sufficient information to allow the system to start calculating the signature before the actual signature packet (which is after the data) is reached. There can be several such packets. One packet contains the public key algorithm, the hashing algorithm, the key ID of the signing key, and an indicator whether the signatures should be nested or not. A zero value indicates that the next packet is another One-Pass Signature packet that describes another signature to be applied to the same message data. **Note** Nested signatures are not supported. However, several signatures over the same data in one PGP message are supported.
Public Key	
Public Subkey	Contains similar information to a public key package, but it denotes a subkey.
Secret Key	Contains all the information that is found in a public key packet, but also includes the secret key (encrypted private key).
Secret Subkey	Contains similar information to a secret key package, but it denotes a subkey.
Compressed Data	Typically, this packet contains the contents of an encrypted packet, or follows a Signature or One-Pass Signature packet, and it contains a literal data packet.
Symmetrically Encrypted Data	Data encrypted with a symmetric key (using a symmetric key algorithm). The symmetric cipher used may be specified in a Public-Key or Symmetric-Key Encrypted Session Key packet that precedes the Symmetrically Encrypted Data packet. This packet uses a variant of the cipher feedback mode (CFB) (as defined at http://tools.ietf.org/html/rfc4880).
Literal Data	Contains plain data (binary or text).
User ID	Indicates the holder of a key. The package contains the user name, e-mail address, and comment of the keyholder.

Packet Type	Description
User Attribute	Variant of the User ID packet, which can contain more information about the user. It is only used together with key material. This packet is not supported.
Sym. Encrypted and Integrity Protected Data	Variant of the Symmetrically Encrypted Data packet. It contains data that is encrypted with a symmetric key algorithm (using a symmetric key algorithm) and is protected against modification by the SHA-1 hash algorithm (less strong than a signature, but stronger than bare CFB encryption). It does not use Open PGP CFB mode but pure CFB mode.

Restrictions for the Input Message Structure (for Decryptor/Verifier)

The input payload must have the following packet sequence:

Public Key Encrypted Session Key ..., Sym. Encrypted and Integrity Protected Data | Sym. Encrypted Data, (Compressed Data,) (One Pass Signature ...,) Literal Data, (Signature ...,)

Entries in brackets are optional, ellipses indicate repetition, commas represent sequential composition, and '|' separates alternatives.

For example, the Compressed Data packet is optional.

Restrictions for the Output Message Structure (for Encryptor/Signer)

The output PGP message is restricted to the following packet sequence:

Public Key Encrypted Session Key ..., Sym. Encrypted and Integrity Protected Data | Sym. Encrypted Data, Compressed Data, (One Pass Signature ...,) Literal Data, (Signature ...,)

Entries in brackets are optional, ellipses indicate repetition, commas represent sequential composition, and '|' separates alternatives.

This does mean the following:

- A symmetric key cannot be encrypted with another symmetric key.
  The symmetric key that encrypts the payload cannot be encrypted by another symmetric key (which is, for example, generated from a password). OpenPGP allows this (see Symmetric Key Encrypted Session Key packet).
- Compression cannot be switched off. The Compressed Data packet is always mandatory.
  However, it is possible to choose the UNCOMPRESSED algorithm. In this case, the Compressed Data packet is still there, but contains the Literal Data uncompressed.
- Encryption is always mandatory. It is not possible to only sign data.
- Only one password for all private keys in the keyring can be used. This simplifies password maintenance.

- Nested signatures are not supported: If there are multiple signatures in the PGP message, they all contain the same hash value built over the original payload. OpenPGP does allow nested signatures where the enclosing signature is a signature of the enclosed PGP message including the enclosed signatures.
- DSA keys can only be combined with certain hash algorithms.

10.1.4 Certificate Management

Depending on the applied transport- and message-level security option, different types of security artifacts need to be managed and deployed on the tenant.

- X.509 certificates
  Used for transport-level security TLS and for message-level security using PKCS#7, WS-Security, and XML Digital Signature.
  They are stored in a Java keystore.
- PGP keys
  Used for message-level security using Open PGP.
- Known hosts files
  Required for transport-level security SFTP.
  SFTP keys are also stored in a Java keystore.

Related Information

X.509 Certificates [page 240]
PGP Keys [page 245]
Known Hosts File [page 245]

10.1.4.1 X.509 Certificates

X.509 certificates (that comply with the X.509 standard) are used for transport-level security TLS and for message-level security using PKCS#7, WS-Security, and XML Digital Signature.

Elements of X.509 Certificates

This topic does not explain the standard in detail, but points out the following important elements of an X.509 certificate.

A digital certificate provides a public key that is signed by a certification authority (CA).

Elements of X.509 Certificates

Element	Description
Issuer	Specifies the CA (that issued and signed the certificate).
Subject	Specifies the entity associated with the public key of the certificate.
Distinguished Name (DN)	Comprises the issuer, the subject, and other attributes. A DN is a unique identifier of the certificate.

When you specify a certificate, you have to define additional attributes such as a company name, a country identification, and so on.

Related Information

Keystores [page 241]
Requirements for Keystore Passwords [page 245]
Certificate Chains [page 244]

10.1.4.1.1 Keystores

X.509 certificates and key pairs are stored in one keystore per tenant.

On each tenant, exactly one keystore needs to be deployed. Therefore, this keystore is also referred to as the *tenant keystore*.

> **Note**
>
> To use X.509 certificates and key pairs, you need to use keystores with the JKS or JCEKS format. We recommend using the `JCEKS` format because it uses a stronger encryption algorithm.

Keystore Usage

A tenant keystore is used to secure message exchange both at transport level and at message level.

- Transport-level security
  Keystores are used in HTTPS outbound connections from the SAP Cloud Platform Integration tenant to a remote system (for example, when using a SOAP or HTTP receiver adapter).
  In such calls, the tenant plays the role of the client and the remote system the role of the server. The trusted certificate entries together with the certificates of the key-pair entries (without the CA certificates of the chains) are used to validate the certificate chain of the server during the TLS handshake. Therefore,

you can control the possible connections to servers with the trusted certificates. You should only add those CA certificates to the keystore that are needed to verify the chains of the servers you want to connect to. It is best practice that the server certificate chain contains the server certificate and the intermediate certificate. That way, the keystore only needs to contain the self-signed root CA certificate as the Certificate entry. The server certificate chain must be verified.
Key-pair entries with their certificate chain are relevant for HTTPS outbound connections with client certificate authentication. During the TLS handshake, one of the key pairs whose certificate chain is trusted by the server is selected for the TLS communication. If the server does not have the appropriate CA certificate in its trust store, the communication fails because the server cannot authenticate the client. If the server trusts several key pairs, one key pair is chosen at random for the connection. If you want to avoid random selection, you can specify an alias of a key-pair entry in the related receiver adapter, so that only this specific key pair can be used in the TLS communication. If the keystore contains only one key pair or the server only trusts one key pair, this measure is not necessary. In some cases it is necessary to adapt the chain of the key pair. For example, if the chain of the key pair contains only the public certificate and the server contains only the root CA certificate, then you need to add the intermediate certificate to the chain of the key pair.
The keystore can also contain additional private keys with the aliases `id_rsa` or `id_dsa` (depending on the key type: RSA or DSA), which are used for the SFTP adapter.

- Message-level security
  The keystore also contains the public and private keys used for message-level security (signing and encryption). Public keys are used in the signature verification steps (XML Signature, PKCS#7/CMS Signature Verification, WebService Security) and in the encryption steps (PKCS#7/CMS, WebService Security) of integration flows. Private keys are used in the signature creation steps (XML Signature, PKCS#7/CMS Signature, WebService Security) and decryption steps (PKCS#7/CMS, WebService Security) of integration flows. In these steps, the relevant keystore entries are referenced by their aliases. We recommend using different keys for message- and transport-level security. Keep in mind that the expiration date of the certificates is not checked in the encryption/decryption steps and in the signing steps.

Keystore Content

There are the following entry types:

- *Key Pair* entry
  Consists of a private key and its X.509 certificate chain.
  All private keys of a keystore are encrypted with the same password. This password is also used as the keystore password (for checking the integrity of the keystore). The keystore is never stored in the same database as the encrypted/signed application data. The password is stored in a separate database.
  The certificate chain typically consists of the public key certificate and the intermediate certification authority (CA) certificate with which the signature of the public key certificate can be verified.
- *Certificate* entry
  In many cases this is an X.509 root certificate.

Keystore Management

A tenant keystore contains both entries owned by the tenant administrator (tenant owner) and entries owned by SAP. SAP-owned entries cannot be changed or deleted by the tenant administrator and entries owned by the tenant administrator cannot be changed or deleted by SAP.

- Keystore managenment by the tenant administrator
  A Keystore Monitor allows tenant administrators to manage their keystore entries and display the SAP-owned entries. The tenant administrator cannot change or delete SAP-owned entries. Likewise, the SAP employee responsible for managing the SAP-owned entries cannot change those owned by the tenant administrator.

  > **Note**
  >
  > This feature is only available for node assembly version 2.29.* and higher.

  Upload, download, and deletion of the keystore entries are restricted by special user roles, which are assigned by default only to the tenant administrator.
- Keystore managenment by SAP
  If SAP manages the keystore entry, it takes special precautions for its maintenance.
  When SAP generates a key pair (consisting of a public key and the corresponding private key) during the setup of a new customer tenant, and subsequently issues a certificate signing request, this all happens within a dedicated secure environment only used for this purpose. Only certain operators at SAP have permission to perform these tasks.

> **Note**
>
> There is a dedicated naming convention for keystore aliases to indicate the owner of the keystore entry:
>
> Alias names of SAP-owned entries start with `sap_` or are `hcicertificate`, `hcicertificate1`, `hcimsgcertificate`.
>
> Exceptions can occur in partner-owned tenants: Here, `hcicertificate`, `hcicertificate1`, and `hcimsgcertificate` can belong to the tenant administrator if they have already been used in a node assembly version prior to 2.29.*.

SAP Cloud Platform Integration does not verify the signatures of the certificates during the upload. Therefore, the user who uploads the certificates is responsible for ensuring that the signatures of the certificates are verified before the upload. Note that root certificates in particular must always be verified manually in any case.

> **Note**
>
> If you are using a node assembly version lower than 2.29.*, be aware of the following implications:
>
> In this case, one party (SAP or the customer) can always download, upload, and delete the keystore entries owned by the other parties. Therefore,strict governance processes must be employed to ensure that only the party that is in charge of managing the keystore entry can use the upload and delete option. Otherwise, one party could overwrite the keystore entry of the other party, which could have severe implications for the message processing and communication setup.

10.1.4.1.2 Certificate Chains

The trust relationship between a client and a server using TLS authentication is usually based on chain certificates.

When using the X.509 standard, a key pair used for the TLS handshake is usually signed by a certification authority (CA). This means that the server can assume that the public key (included in the certificate) provided by the client originates from a trusted source.

The X.509 standard allows you to build up hierarchical trust models. In such a model (also referred to as a *certificate chain*), many certification authorities (CAs) are involved on different hierarchy levels. This means that the certificate that identifies the CA as a trusted participant can itself be signed by a CA at a higher level in the hierarchy. This means that a number of (intermediate) CAs can be arranged above the actual client certificate. The highest level CA is called the root CA.

The following figure shows a certificate chain with two intermediate CAs:

We assume that the tenant is connected as a client to an external component (which can be referred to as the server or receiver system).

To establish SSL connectivity, the server is provided with the root CA certificate and nothing else. To make sure that a trust relationship between client and server can be established nevertheless, the client certificate (of the tenant) used for the SSL handshake has to contain the whole certificate chain. In other words, the client certificate has to include all intermediate CAs (excluding the root CA). This enables the server to evaluate and calculate the whole chain of trust.

Therefore, during connection setup (onboarding), the tenant key pair (client certificate) has to be assigned the whole certificate chain.

> **Tip**
>
> To find out the certificate chain of the server, you can use the TLS Outbound Connection Test (accessible in the Monitoring application). This test also helps you to find out whether you have the correct CA certificate in the keystore to validate the server certificate chain (see option *Validate Server Certificate* of the Outbound Connection Test).

Related Information

TLS Connectivity Test [page 127]

10.1.4.1.3 Requirements for Keystore Passwords

To protect a keystore, you have to specify a password when creating the keystore.

You have to apply the following rules when specifying passwords for keystores:

- The password must have a minimum length of 8 characters.
- The password must contain characters of **at least three** of the following groups:
  - Lower-case Latin characters (a-z)
  - Upper-case Latin characters (A-Z)
  - Base 10 digits (0-9)
  - Non-alphabetic characters (!@#$%...)
- The password must **not** contain any characters from outside the standard ASCI table like, for example, German umlaut characters (`<ü>`).

> **Note**
>
> Example for password compliant with the above rule:
>
> `<xB+gku!kjhz>`

10.1.4.2 PGP Keys

PGP public and secret keys (the latter containing a private key) can be uploaded to the tenant via separate keyrings. The PGP Public Keyring contains Transferable Public Keys as defined in section 11.1 of the Open PGP specification (https://tools.ietf.org/html/rfc4880) and the secret keyring contains Transferable Secret Keys as defined in section 11.2.

PGP keys are used in the PGP Encryptor and Decryptor step. You should only add PGP Public keys to thePGP Ppublic Keyring if you trust this key. Typically you check the fingerprint of the public key. The same security measures must be taken for the secret keys which you use in the secreet keyring. The encryption and signing steps do also work with expired certificates.

For the PGP Secret Keyring the same precautions as for the X.509 keystore must be taken because it contains private keys.

10.1.4.3 Known Hosts File

Known hosts files are relevant for SFTP communiction. The known hosts file contains the host names and the public keys of the trusted SFTP servers. You should only have entries for those serves in the file which are used by the integration flows of the tenant and which you trust.

10.2 Security Elements

To set up the secure communication between a tenant and a sender/receiver system, certain security elements have to be created and - in some cases - exchanged between the involved components (the tenant on the one side and the sender/receiver system on the other side of the communication).

For example, to set up SSL communication using certificate-based authentication between a tenant and a receiver system, X.509 certificates are required. Those private keys owned by the tenant are to be part of a Java keystore that is to be deployed on the tenant, whereas the private keys owned by the receiver are to be part of the receiver system keystore. To complete the security setup, each keystore also has to contain the public key of the connected partner. In our example, the Java keystore of the tenant has to contain the receiver public key, and the receiver keystore has to contain the tenant public key.

This section provides a summary for each security option of how the required security elements have to be distributed among the involved components (tenant and sender/receiver systems).

Related Information

Security Elements (Transport-Level Security) [page 246]
Security Elements (Message-Level Security) [page 249]
How Security Artifacts Are Related to Integration Flow Configuration [page 253]

10.2.1 Security Elements (Transport-Level Security)

Each transport-level security option requires a specific set of security elements.

The following tables provide a summary of how the required security elements (in **bold letters**) have to be distributed among the involved components (tenant and sender/receiver systems).

Transport-Level Security

Security Option	Direction	Required by tenant administrator ...	... to do the following	Required by sender/receiver administrator ...	... to do the following
HTTPS – basic authentication	Inbound (sender calls tenant)	**User name** of SAP Cloud Platform (to be provided by sender administrator). This is the user under which the customer system is to call the integration platform of SAP Cloud Platform.	Grant the required authorizations to enable this user to call the tenant.	**Load balancer root certificate** (to be provided by tenant administrator) Is required for the SSL communication step (can be obtained via the URL of the runtime node provided in the tenant mail by SAP).	Import into the keystore of the sender system.
				User name and password (to be provided by tenant administrator)	Enable the sender to support basic authentication.
	Outbound (tenant calls receiver)	**Receiver server root certificate** (to be provided by receiver administrator) Is required to enable HTTPS communication with the receiver system (server).	Import into the tenant keystore (and deploy the keystore on the tenant).		
		User credentials (to be provided by receiver administrator) These are the user credentials under which the tenant is to call the receiver system.	Define the User Credentials artifact (to be deployed on the tenant).		

Security Option	Direction	Required by tenant administrator ...	... to do the following	Required by sender/receiver administrator ...	... to do the following
HTTPS – certificate-based	Inbound (sender calls tenant)	**Sender client root certificate** (to be provided by sender administrator)	Check whether the CA the customer system used to get its client certificate signed is already part of the load balancer (server) keystore.	**Load balancer server root certificate** (to be provided by tenant administrator)	Import into client PSE of the sender system.
		Sender client certificate (to be provided by sender administrator)	Configure the authorization check in the integration flow.		
				List of trusted root certificates supported by load balancer (to be provided by tenant administrator)	Select a certification authority from the list for the certificate signing request for the client certificate.
	Outbound (tenant calls receiver)	**Receiver server root certificate** (to be provided by receiver administrator)	Import into tenant keystore (if not already there).	**Tenant client root certificate** (to be provided by tenant administrator)	Import into the server PSE of the receiver system.
				Tenant client certificate (to be provided by tenant administrator)	Define the client certificate-to-user mapping for the configuration of authorization checks.
SFTP	Outbound (tenant as SFTP client sends a request to an SFTP server)	**SFTP server (receiver) public key** (to be provided by SFTP server (receiver) administrator) Is required by tenant to check whether SFTP server can be trusted.	Add to known_hosts file (to be deployed as Known Hosts artifact on tenant).	**Tenant public key** (to be provided by tenant administrator) Is used to authenticate tenant as a trusted SFTP client on the SFTP server side.	Add to authorized_keys file on the SFTP server side.

10.2.2 Security Elements (Message-Level Security)

The configuration of secure message exchange requires the exchange of public keys (or other security-related information) between the involved parties. Each message-level security option requires a specific set of keys to be exchanged.

The following tables provide a summary of how the required security elements (in **bold letters**) have to be distributed among the involved components (tenant and sender/receiver systems).

Message-Level Security

Security Option/Standard	Direction	Protection Method on Tenant	Required by tenant administrator ...	... to do the following	Required by sender/ receiver administrator ...	... to do the following
PKCS#7, WS-Security, XML Digital Signature (uses X.509 certificates) XML Digital Signature: only sign/encrypt	Inbound (sender calls tenant)	Decrypt			**Tenant public key certificate** (to be provided by tenant administrator) Is used to encrypt the message from the sender (that is to be encrypted by the tenant).	Import into sender keystore
		Verify	**Sender public key certificate** (to be provided by sender administrator) Is used by the tenant to verify the signature of the message sent from the sender system.	Import into tenant keystore.		
	Outbound (tenant calls receiver)	Encrypt	**Receiver public key certificate** (to be provided by receiver administrator) Is used by the tenant to encrypt the message (sent to the receiver).	Import into tenant keystore.		

Security Option/Standard	Direction	Protection Method on Tenant	Required by tenant administrator ...	... to do the following	Required by sender/ receiver administrator ...	... to do the following
		Sign			**Tenant public key certificate** (to be provided by tenant administrator) Is used by the receiver to verify the message sent from the tenant.	Import into receiver keystore
OpenPGP (uses PGP keys)	Inbound (sender calls tenant)	Decrypt			**Tenant public key** (to be provided by tenant administrator) Is used to encrypt the message from the sender (that is to be encrypted by the tenant). To make sure that the public key originates from the correct source and that it has not been changed on its way, consider the note below this table.	Import into sender PGP public keyring

Security Option/Standard	Direction	Protection Method on Tenant	Required by tenant administrator ...	... to do the following	Required by sender/ receiver administrator ...	... to do the following
		Verify	**Sender public key** (to be provided by sender administrator) Is used by the tenant to verify the signature of the message sent from the sender system. To make sure that the public key originates from the correct source and that it has not been changed on its way, consider the note below this table.	Import into tenant PGP public keyring.		
	Outbound (tenant calls receiver)	Encrypt	**Receiver public key** (to be provided by receiver administrator) Is used by the tenant to encrypt the message (sent to the receiver). To make sure that the public key originates from the correct source and that it has not been changed on its way, consider the note below this table.	Import into tenant PGP public keyring.		

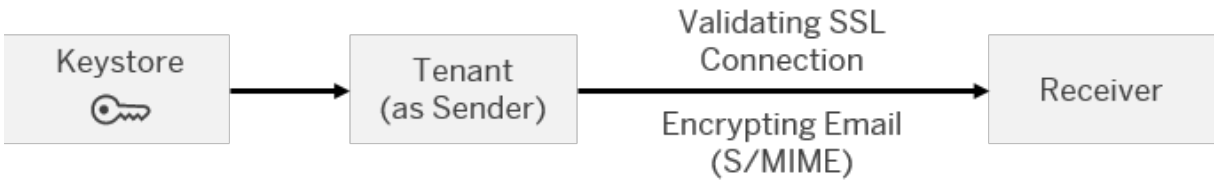
Security Option/Standard	Direction	Protection Method on Tenant	Required by tenant administrator ...	... to do the following	Required by sender/ receiver administrator ...	... to do the following
		Sign			**Tenant public key** (to be provided by tenant administrator) Is used by the receiver to verify the message sent from the tenant. To make sure that the public key originates from the correct source and that it has not been changed on its way, consider the note below this table.	Import into receiver PGP public keyring

> **Note**
>
> Relevant for the SAP-managed operating model: When **exchanging public PGP keys**, note the following:
>
> To ensure that the information originates from the correct source and that it has not been changed on its way, the key should be exchanged using a secure channel (for example, encrypted e-mail).
>
> If a secure channel is not available, the person who receives the public key from the key owner has to **verify the fingerprint** of the public key. One option is to phone the owner of the public key and compare the fingerprint.

10.2.3 Security Elements Related to the Mail Adapter

The usage of the mail adapter requires certificates both to validate the SSL connection and to encrypt the mail (in case S/MIME has been chosen).

The sender mail adapter enables the tenant to send an (encrypted) email to a receiver system, as illustrated in the following figure.



The tenant keystore needs to contains the following certificates:

Certificates to be Importted into Tenant Keystore

Certificate	Purpose
Receiver server root certificate	For SSL connection: This certificate is required to identify the root CA that is at the top of the certificate chain that finally guarantees the trustability of the receiver server certificate.
Tenant client certificate	For SSL connection: This certificate is required to authenticate the tenant when calling the receiver system as client.
Public key (selected according to public key alias name configured in mail sender adapter)	This certificate is required to encrypt the email.

10.2.4 How Security Artifacts Are Related to Integration Flow Configuration

To specify the security-related aspects of the message flow, certain settings have to be made in the involved integration flows. These security settings are related to the security artifact deployed on the involved tenant.

The following example gives you an idea of how security artifacts and integration flow settings are related to each other: In order to specify in detail how a message is to be digitally encrypted, you need to define an Encryptor step in the relevant integration flow. At runtime, this Encryptor step needs to access the required public key to encrypt the message content. The public key itself has to be available in the keystore that is deployed on the involved tenant.

This section summarizes the following information for each security option:

- The required security artifact type (to be deployed on the tenant)
- The required step or adapter type (relevant for the related integration flow design)

Transport-Level Security Key Types

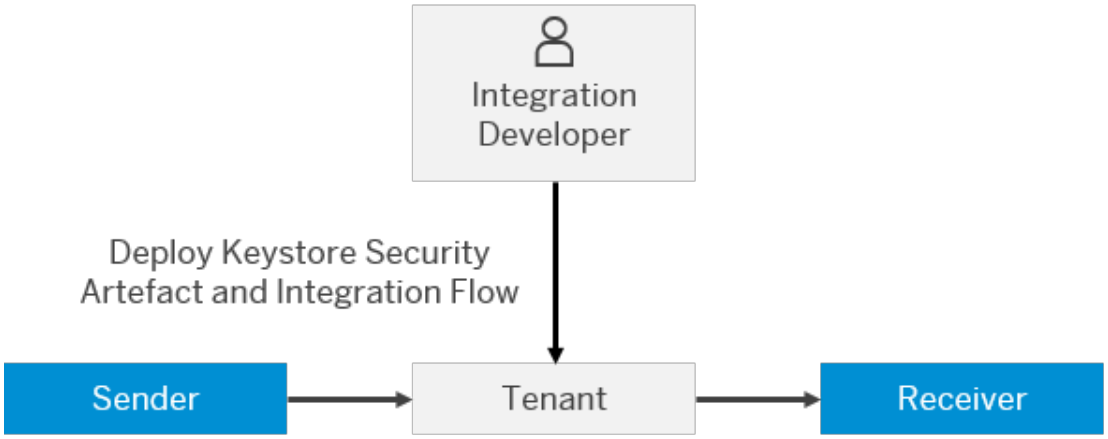
Transport-Level Security	Key Type	Artifact Type (to Deploy on Tenant)	Integration Flow Step/ Adapter Type
HTTPS - Basic Authentication	User credentials (user name and password)	User Credentials	SOAP adapter, IDoc adapter, HTTP adapter, SuccessFactors adapter

Transport-Level Security	Key Type	Artifact Type (to Deploy on Tenant)	Integration Flow Step/ Adapter Type
HTTPS (SSL) - Certificate-Based Authentication	X.509 certificates	Keystore	SOAP adapter, IDoc adapter, HTTP adapter, SuccessFactors adapter
SFTP (SSH)	SFTP key and known_hosts	Keystore Known Hosts	SFTP adapter

Message-Level Security Key Types

Message-Level Security	Key Type	Artifact Type (to Deploy on Tenant)	Integration Flow
PKCS#7	X.509 certificates This is the same key type as used when setting up HTTPS-based transport-level security. When setting up the security level for a tenant, you can use the same keystore for transport-level security keys (if you are setting up HTTPS-based communication) and for message-level security keys. Note, however, that we recommend **using different keys** for transport-level and message-level security.	Keystore	Signer, Encryptor, Verifier, Decryptor
XML Digital Signature			Signer, Encryptor, Verifier, Decryptor
WS-Security			SOAP adapter (SOAP 1.x)
OpenPGP	PGP public keys and PGP secret keys	PGP public keyring PGP secret keyring	Signer, Encryptor, Verifier, Decryptor

The following figure illustrates the overall setup with regard to the tenant (for a situation where a keystore containing a public-private key pair is deployed on the tenant as a security artifact).


Integration Developer
Deploy Keystore Security Artefact and Integration Flow
Sender
Tenant
Receiver

Important Disclaimers and Legal Information

Coding Samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP's gross negligence.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with "you", or a gender-neutral noun (such as "sales person" or "working days") is used. If when referring to members of both sexes, however, the third-person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP's gross negligence or willful misconduct. All links are categorized for transparency (see: https://help.sap.com/viewer/disclaimer).

go.sap.com/registration/contact.html